A Conversational Agent for Dynamic Procedural Interactions

by

Pedro Colón-Hernández

S.M., Massachusetts Institute of Technology (2018)
B.S., University of Puerto Rico at Mayagüez (2016)

Submitted to the Program in Media Arts and Sciences
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Media Arts and Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

©2023, Pedro Colón-Hernández, All rights reserved
The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Author:

Pedro Colón-Hernández
Program in Media Arts and Sciences
May 19, 2023

Certified by:

Dr. Cynthia Breazeal
Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by:

Tod Machover
Academic Head, Program in Media Arts and Sciences

A Conversational Agent for Dynamic Procedural Interactions

by

Pedro Colón-Hernández

Submitted to the Program in Media Arts and Sciences
School of Architecture and Planning
on May 19, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Media Arts and Science

Abstract

How-To questions (e.g., "How do I cook rice?", "How do I write a check?", or "How do I send pictures to my family from my iPhone?") are some of the most common questions asked of search engines [37] and presumably of conversational agents as well. Answers to How-To questions should generally be in the form of a procedure; step-by-step instructions that users perform in sequence. However, people find reading instructions cognitively demanding and often prefer that another person guide them through a procedure [44]. Prior work in automating procedural guidance either concentrates on how to communicate instructions or how to reason about procedural knowledge to extract states of entities. In this work, we present an end-to-end procedural voice guidance system that automatically generates and presents step-by-step instructions to users through a conversational agent. This system overcomes three significant challenges: generating a contextual knowledge graph of the procedure, ordering necessary information through reasoning on that graph and converting it to procedural steps, and finally constructing a conversational system that delivers the procedure in a way that is easily followed by users. Our approach improves upon the current state-of-the-art in conversational agents, which often hand off the interaction to a web search. We demonstrate that our system can be utilized for end-user guidance, and that a contextual commonsense inference system can be used for procedural knowledge graph generation and ultimately procedural step generation. We also show that reasoning for procedural step generation is essential for the task. Lastly, we show that combining our knowledge driven system, both its steps and contextual commonsense assertions with a large language model (LLM) provides more accurate and reliable procedural guidance in tasks that the LLM may have trouble recalling/or were created after training. This work opens up paths to perform contextual graph-based reasoning for story-based applications and helps inform the design of future conversational agents within the domain of procedural guidance.

Thesis Supervisor: Dr. Cynthia Breazeal
Title: Professor of Media Arts and Sciences

Acknowledgments

I would like to thank everyone that has helped me in the past seven years; and believe me when I say, this has been a LOT of people. I'd like to start thanking my family: my mom Eva and my dad Ricardo, who have always been there to talk to me on the phone whenever things have gone well or when things have gone bad. I'd like to thank my brother Ricardo and my sister-in-law Cynthia, who have been with me in this process since before I even started. They have helped me move a record $\approx$7 times across the country and within the state and have been there also in the good, the bad, and the crazy times. I would also like to thank my twin Eva, who has also given me support, laughs, and has come with me on various trips throughout these years.

I would also like to thank Cynthia, my PI, for giving me a chance to be a part of the group and giving me the opportunity to pursue my dream research. I would like to thank Henry and Philippe, who have been instrumental in shaping my career, research, and experiences throughout. I would like to thank Bob for answering a cold email for being a dissertation reader/advisor and helping me even though you barely knew me. I would like to thank Yejin for advising and connecting me with such nice collaborators such as Saadia and Liwei. I would like to thank Sharifa, Hae Won, and all the research staff in PRG for giving help and guidance when needed throughout these years. I would also like to thank Randi, who is the person I have known for the longest, for helping me when my previous group (Object Based Media) dissolved, and for always being unconditionally there whether to get a coffee, talk, or just walk. I would like to thank all the robots that have helped make my time in this process be enjoyable, Daniela, Safinah, Brayden, Jocelyn. I would also like to thank the members of my old lab group OBM, Pip, Laura, Sunny, both Emily's, Nina, and Bianca who have all been instrumental in me making it to the lab and making me feel at home in the lab. I would like to thank our admins Polly, Kristin, and Amna for being able to do miracles to make our life easier as students. I would also like to thank Samantha and Alessandra, for being there to get coffee, dinner, and everything

in between during all these years.

Now I'd like to thank the friends that have been with me since before this process and the ones that I met while going through it. In no particular order; I'd like to thank Antoine and Marie Ann who I've known for almost half of my life and have both joked with and given support in many times. I'd also like to thank Christian and Tatiana. I don't think I would be here if you had not reached out to work together in a project $\approx$10 years ago, and I can't believe that we have gone through so many changes in life together and apart. I would also like to thank Strong Jean (yeyeye) who has consoled me in constructive and destructive ways throughout my time in Cambridge/Boston and has helped me get out of so many holes I have dug myself into. I would also like to thank Juan, who helped me survive the pandemic with an excess of Domino's and laughter. I would like to thank Manuela and Paola, who, although they live in New York, would always be down to hang or visit me throughout all these years. I'd like to thank Leo, both Lauras, Nina and Nino, Omar, Gustavo and Krisia, Juan Ma, Marisel, Karla, Francine, Juan and Larimar, Baldin and Ivys, and many more people who collectively in group chats or in Perdicion helped me also feel at home and connect with friends when I started graduate school and throughout it. I would like to thank Caroline and Andrea for making me learn, grow, explore and live. I'd like to thank Yessika for being so kind, helpful, and loving through the end of this process. I don't think I could have finished if I did not have your help and support.

Lastly, I would like to thank the hundreds if not thousands of people that I have met throughout this process. Whether at parties, conferences, classes, the lab, in Cambridge or in Boston, in Massachusetts, California, New York, or Puerto Rico, for helping me get through this process and come out on the other side as a better person.

Thank you all! I dedicate this thesis to all those people who have motivated this work by calling me as their "tech-support". I hope that now you can utilize one of these agents to get "tech-support".

This doctoral thesis has been examined by the following committee:

Dr. Cynthia Breazeal

Professor of Media Arts and Sciences
Media Lab

Dr. Henry Lieberman

Research Scientist
MIT CSAIL

Dr. Yejin Choi

Brett Helsel Professor
University of Washington

Dr. Robert J. Moore

Research Staff Member
IBM Research

Dr. Philippe Piernot

Conversational Prototyping Manager
Apple, Inc.

Contents

1 Introduction **45**

1.1 Preamble 45

1.2 Motivation 45

1.3 Overview 49

  1.3.1 Abstracting procedures to knowledge graphs 49

  1.3.2 Reasoning over procedural knowledge graphs for step generation 50

  1.3.3 Conversational agent and patterns for communicating steps . . 52

  1.3.4 Guiding Example 53

1.4 Research Questions and Novelty 55

  1.4.1 Research questions 55

  1.4.2 Novelty 57

1.5 Contributions 59

2 Background **61**

2.1 Procedures and Instructions 61

2.2 Knowledge Graphs 63

  2.2.1 General Knowledge Graphs 63

  2.2.2 Commonsense Knowledge Graphs 64

2.3 Transformers and Natural Language Processing 66

  2.3.1 Very Large Language Models 69

3 Related Work **71**

3.1 Relevant work for Abstracting Procedures into Facts 71

3.1.1 Large Language Model Prompting 71
3.1.2 Controllable Generation 72
3.1.3 Story and Assertion Alignment 73
3.2 Relevant work for Processing Abstracted Procedural Knowledge into Steps 74
3.2.1 Text Generation and Reasoning with Knowledge Graphs . . . 74
3.3 Relevant work for Relaying Steps through Conversational Agents . . . 77
3.3.1 Modern Collaborative Agents 77

4 Abstracting Procedures into Facts 85
4.1 Overview 85
4.2 Contextual Knowledge Graph Generation 85
4.3 Hinting for Controllable Generation 91
4.3.1 What exactly is hinting? 91
4.3.2 Discourse-aware/contextual commonsense inference 93
4.3.3 Testing Hinting: Experimental Setup 94
4.3.4 Results and Effects of hinting 97
4.3.5 Final Remarks on Hinting 98
4.4 Joint Inference of Commonsense Assertions 99
4.4.1 What is joint commonsense inference? 99
4.4.2 Joint Inference Approach Overview 100
4.4.3 Specificity in assertions 101
4.4.4 Aligning Assertions with Stories 103
4.4.5 Experimental Setup 104
4.4.6 Effects of joint inference 106
4.5 Adversarial Language Models 109
4.5.1 Joint and Adversarially training language models 109
4.5.2 Addressing the Discontinuity in Generation 111
4.5.3 Addressing Different Generation Types 113
4.5.4 Splicing different models 114

4.5.5 Factuality in the Discriminator 114
4.5.6 Experimental Setup 115
4.5.7 Effects of adversarial training 115
4.6 Contextual Commonsense Inference Summary 116
4.7 Procedure Ingestion and Abstraction System 117
4.7.1 Procedural Abstraction Overview 117
4.7.2 wikiHow as a Procedure Dataset 118
4.7.3 When retrieval doesn't match: dynamic lookup and extraction 122
4.7.4 Contextual Procedural Knowledge Generation 127
4.8 Summary 129

5 Processing Abstracted Procedural Knowledge into Steps 131
5.1 Overview 131
5.2 Knowledge Driven Procedural Step Generation 132
5.2.1 Problem Statement 132
5.2.2 Procedural Step Realization Model 133
5.2.3 Dataset 134
5.2.4 Experiments, Results, and Analysis 136
5.2.5 Conclusions 142
5.3 Reasoner 143
5.3.1 Reasoner Models 143
5.3.2 Conclusions 151
5.4 Procedure Error Handling 152
5.4.1 Algorithm 152
5.5 Summary 154

6 Relaying Steps through Conversational Agents 157
6.1 Overview 157
6.2 Conversational Patterns 158
6.3 Conversational Interaction Considerations 164
6.3.1 Simple, user-oriented language 164

6.3.2 User control 164
6.3.3 Consistency in platform 165
6.3.4 Error prevention and handling 165
6.3.5 Recognition rather than recall 165
6.3.6 Minimalistic design or principle of minimization 165
6.3.7 Error messages 166
6.3.8 Help and documentation 166
6.3.9 Conversation first rather than visual, system, or content first . 166
6.3.10 Content formatting 167
6.3.11 Basic conversational functionality 167
6.3.12 Consistent persona 167
6.3.13 Feedback & Responsiveness 168
6.4 Conversational System for Procedural Guidance 168
6.4.1 Approach Overview 169
6.4.2 Architecture 170
6.4.3 Intent-Based Procedural Guidance 177
6.5 Large Language Model-Based Guidance 179
6.5.1 Plain, zero-shot guidance agent 179
6.5.2 Hybrid: Knowledge Driven LLM 180
6.6 Summary 183

7 Evaluating a Conversational Agent for Procedures 185
7.1 Overview 185
7.2 Evaluation 1: Task Completion 186
7.2.1 Overview 186
7.2.2 Research Questions 186
7.2.3 Study Design 187
7.2.4 Study Results 191
7.2.5 Overall Findings 218
7.3 Evaluation 2: Error Recovery 219

7.3.1 Overview . . . 219

7.3.2 Research Questions . . . 219

7.3.3 Study Design . . . 220

7.3.4 Study Results . . . 222

7.3.5 Overall Findings . . . 245

7.4 Evaluation 3: Leveraging both flexibility of LLMs and knowledge driven steps . . . 247

7.4.1 Overview . . . 247

7.4.2 Research Questions . . . 247

7.4.3 Study Design . . . 247

7.4.4 Study Results . . . 249

7.4.5 Overall findings . . . 257

7.5 Future Guidelines . . . 258

7.6 Limitations in Studies . . . 259

7.6.1 Reading Instructions from a Web-page . . . 259

7.6.2 Lack of usage tutorial . . . 260

7.7 Summary . . . 260

8 Limitations and Discussion 263

8.1 Expert-Domain Guidance . . . 263

8.2 Reasoning Assumptions . . . 265

8.3 Intent recognition systems . . . 266

8.4 Large Language Models - Lack of conversational patterns . . . 267

8.4.1 Outdated training data . . . 274

8.5 Explainability . . . 278

8.6 Summary . . . 280

9 Contributions and Future Work 281

9.1 Future Work . . . 281

9.1.1 Embodiment . . . 281

9.1.2 Visual Outputs for Guidance . . . 282

9.1.3 Multi-modal Inputs for Guidance 282

9.2 Final Remarks 284

9.2.1 Summary of Contributions 288

A Study Questionnaires 291

A.1 Study 1 291

A.2 Study 2 311

A.3 Study 3 338

B Additional 363

B.1 Cosine Reasoner 363

B.2 Existing Procedural Interaction: Cooking 364

B.2.1 Heuristic Evaluation 364

B.2.2 Small User Study for Usability of the Agents 365

B.3 Additional Studies 366

B.4 Long-Term Open-domain Guidance 366

B.4.1 Overview 366

B.4.2 Study Design 366

B.4.3 Study Results 367

B.4.4 Overall Findings 373

B.5 Closed Domain (in-person) Guidance 374

B.5.1 Overview 374

B.5.2 Study Design 374

B.5.3 Study Results 374

B.5.4 Overall Findings 376

C Related Works 379

C.1 Related Work for Abstracting Procedures to Knowledge Graphs 379

C.1.1 Commonsense: Grounding, Reasoning, and Knowledge 379

C.1.2 Adversarial Language Models 381

C.2 Related Work for Reasoning and Generation of Procedural Steps . . . 381
    C.2.1 Procedural Knowledge Extraction and Understanding 381
    C.2.2 Commonsense and Procedures 385
    C.2.3 Procedure Representations 387
    C.2.4 Combining structured information and language models 390
    C.2.5 Knowledge Graph Reasoning 393
C.3 Additional Work for Text Generation from Knowledge Graphs 396
    C.3.1 Additional 396
C.4 Related Work for Conversational Agents 397
    C.4.1 Classical Collaborative Agents 398

List of Figures

1-1 Behavior of Siri when asked for guidance . The assistant falls back to a web search for the topic and gives no guidance. 47

1-2 Behavior of Google Assistant when asked for guidance. The assistant gives a copied and pasted response from Wikihow as guidance. 47

1-3 Overview of the process for generating contextual procedural knowledge graphs and steps. From left to right, a given text (e.g., story or procedure description) is iterated sentence by sentence. For each sentence, we provide a hint of what a contextual commonsense inference model should infer a fact about. We then collect these facts for a contextual knowledge graph of the input text. 50

1-4 Overview of the process for generating contextual procedural knowledge graphs for error handling. Similarly to the previous figure, the controllable commonsense inference model is used to generate the contextual knowledge graph of the procedure to perform error handling. However, to generate a step for the procedure, the knowledge graph from the prior goal is utilized in addition to the one for the error. . . 51

1-5 Extended telling conversational pattern. Taken from [108] 52

1-6 Extended telling example. Taken from [108] 52

1-7 Example of Graphical User Interface (GUI) utilized to communicate with our conversational agent. The outputs of the agent are relayed to users through speech-to-text, and the user communicates with the agent by pressing the microphone button and recording a message. . . 53

2-1 Sample sub-graph from ConceptNet [92] 65

2-2 Sample sub-graph from ATOMIC [155] 65

2-3 Sample data from GLUCOSE[113]. The sentence in the gray box is the assertion. We can convert these assertions similar to how ATOMIC converts its assertions as seen in figure 2-2 65

2-4 Simplified overview of the encoder-decoder transformer model. Cutouts and overall architecture from [171] 67

3-1 Procedure utilized by [110] to translate a an input graph into natural language text 75

3-2 Architecture for "Enabling Interactive Answering of Procedural Questions" [99] 78

3-3 Architecture for CARAGS [176] 79

3-4 Example of MAT agent guidance from [179] 80

3-5 Architecture for [118] 81

4-1 Overview of the task of contextual commonsense inference. From the story on the left, and the **bolded** sentence, a model should infer assertions such as the ones on the right. 86

4-2 Step 1: The story and assertion corpus are vectorized. In our work we utilize the sentence-transformers package [146] to achieve this. 104

4-3 Step 2: The resulting assertion vectors are utilized as queries, and the resulting story vectors are used as keys for a memory-like lookup. In this work, we use the FAISS package for this. The output of the memory-like lookup is the nearest story for each vector. This process is repeated for the sentences in the nearest story, to align the assertion with a sentence. 104

4-4 Overview of the proposed GAN architecture. A story and a target sentence are fed into the generator, which infers a contextual commonsense fact. This fact, along with the story and target sentence, is passed into a discriminator to determine whether it is from a generator or not and whether it is factual or not. 110

4-5 We visualize an example that shows the discontinuity when combining a generative language model with a discriminative language model. The dashed line represents where the gradients are discontinued because of the non-differentiable *argmax* operation. 112

4-6 We visualize an example that shows how we address the discontinuity by replacing the non-differentiable *argmax* with a dot product between the softmax and the embedding layer matrix. Additionally, we highlight where the scaling factor is inserted to make the approximation more accurate. We mark our approach in green. 112

4-7 Process to ingest and abstract a procedure. We go from a procedure title, to a document that contains the procedure, to steps that are extracted from the procedure, and finally into facts that are generated from a hinted controllable commonsense inference system. The original data, the steps, and the extracted commonsense assertions are all stored in ElasticSearch. 118

4-8 Web page view for wine opening procedure. Note that above it there is more text and below there are more steps and images. 124

4-9 Readability view for wine opening procedure. Note that the advertisements from above have been removed, however there is still extra text above the procedure. 125

4-10 Visualization of a LLM being utilized for step extraction. Note that the LLM removed unessential text from the procedure, and ignored useless text. 126

4-11 Procedure for abstracting Step 1 in the procedure how to write a check. We can see the extracted noun/verb phrases and how these are used in permutations with relations to produce hints to guide the generation. We note that this is a small example of the contextual knowledge graph for Step 1, and that there are more assertions that are generated from the permutations. 127

5-1 Visualization of process to generate contextual commonsense assertions. Note that the extraction systems receive *all* of the procedure's text including the title to generate keywords/phrases from, and this is what is utilized to do contextual commonsense inference. 135

5-2 Validation results for tests of whether knowledge was present or not. We can see that without knowledge, the model's performance stays almost constant on automated metrics. However, we see that with some noisy knowledge, the performance of the model slowly improves. We see that with filtered (i.e. more relevant/grounded) knowledge, the model performs considerably better. 137

5-3 Training loss plots for varying amounts of commonsense assertions. We see that typically, the more assertions that are given, the lower the loss will be, indicating that more knowledge in the task tends to make it easier. 138

5-4 Validation BLEU results for varying amounts of information. We see that performance keeps increasing as we increase the amount of information. Interestingly, the performance with 48 facts tends to be at first better than that of 64. We see that at the final epochs, the performance of 64 narrowly beats that of 48. In the tables that we present, this does not seem apparent. 139

5-5 Validation metric results for different ordering schemes. We see that the rouge sorting gives the best performance, as it is a metric that can see the output, however the plausibility-based scoring gives the second best ordering. 141

5-6 Validation results of complete model training. We can see that by supplying useful information and enough procedures, even on the first epoch we achieve good performance on a withheld validation set. . . . 142

5-7 Here we see the training metrics for the approximation for the rouge score filtering. We see that the F1 performance plateaus, similar for the MSE loss. 144

5-8 Here we see the validation metrics for the approximation for the rouge score filtering. We see that the F1 performance plateaus, similar for the MSE loss. 144

5-9 Objective function for the unsupervised planner in [183]. We see three terms, the first being the language modeling loss of the system, the second being the term that gets maximized which is the likelihood of the plan given the likelihood of the generated text, and the third an exploration term which is if the likelihood of the generated text has not been maximized, there may be an unseen plan that may be explored.147

5-10 Performance of reinforcement learning planner with frozen, pre-trained realizer. The system somewhat inconsistently improves in performance 148

5-11 Performance of reinforcement learning planner with frozen, pre-trained realizer, for more data and a longer amount of epochs (15 vs. 5). The system somewhat inconsistently improves in performance 149

5-12 Here we can visually see how a sub-procedure exception can be handled. It can be handled either with a simple clarification question which lets a user proceed to the next step, or with a sub-procedure to fix the error. In the case of the sub-procedure, it may also be possible to complete the process in an attempt to fix an error. This can be seen as an alternative way of doing the procedure, and is the reason why there is a dashed line (possible discontinuity) in the end of the exception process. 153

6-1 Architecture used to power conversational agent for procedures. Users interface with a Graphical User Interface (GUI) which lets them talk or type to an agent. The interface pings the backend server for a selected model (Baseline-Gold Instruction Reading Model, KDPSG-Knowledge Driven Procedural Step Generation Model, or GPT- GPT-3 fully driven agent) and gets a response which it then converts to audio. The user receives the textual response and the audio one (if audio is enabled) 171

6-2 The GUI consists of two main screens, the login screen (pictured here) and the conversation screen (pictured below). In the login screen, users can register to have access to a voice assistant or can log in with a registered account 172

6-3 In the chat screen, after logging in, is where a user would communicate with the agent. Here, a user can either dictate (by pressing the microphone button) or type their responses to the agent. The user can also mute the agent by pressing the sound button at the top right . . 173

7-1 User distribution of familiarity with conversational agents. We see most people are at least moderately familiar with conversational agents.192

7-2 Distribution of participant's perception of general usefulness of conversational agents. We see that most people find the agents useful. . . . 192

7-3 Distribution of participants reported interaction quantities with conversational agents. We see that more than half of people use an agent at least once a day or more. 192

7-4 Distribution of participants reported monthly performance of tasks. We see that almost all the selected procedures fell into the "rarely" category (value=1), with only the composing an email procedure being the one that people perform the most. 193

7-5 Distribution of the procedures that were assigned to participants. We see that this follows an inverse distribution as the task ranking, where the procedures that were ranked as most familiar are the least that we select for participants. 194

7-6 Distribution of whether a participant was able to accomplish the goal they were given or not. We see that most of the participants succeeded in this. 196

7-7 Distribution of whether a participant was able to accomplish the instructions that the agent gives. We see that most participants were able to follow what the assistant guided them on. 196

7-8 Example of a catastrophic transcription error that blocked a user. . . 197

7-9 Example of successful intent matching and guidance. 198

7-10 Example of failed intent matching. Examples originally did not include the phrasing "set up [PROCEDURE NAME]". 198

7-11 Distribution of whether an agent gave an appropriate amount of information or not. We see that most people were neutral. 199

7-12 Distribution of whether a participant found the agent conversational or not. Most people did not think the agent was conversational. . . . 199

7-13 Distribution of whether an agent was easy to use or not. Most people did not find the agent easy to use. 200

7-14 Distribution of whether a participant would use the agent frequently or not. Most people would not use the agent frequently. 200

7-15 Distribution of whether the conversation felt natural or not. Most people thought the conversation was not natural. 201

7-16 Distribution of whether a participant had to use special phrasing or not. Most people had to use special phrasing to communicate effectively. 201

7-17 Distribution of whether participants would learn to use the agents quickly or not. Most people would not learn to use the agent quickly. 202

7-18 Distribution of whether an agent gave relevant information or not. Most people were neutral on the agent giving relevant information. . 202

7-19 Distribution of whether participants thought that the agent understood them and helped them achieve their goals. Most people leaned towards thinking that the agent did not understand them. 203

7-20 Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people were neutral on whether the agent was too complex or not. 203

7-21 Distribution of participants' familiarity with conversational agents.Most people are moderately or more familiar. 205

7-22 Distribution of participant's perception of conversational agents. Most people think they are moderately useful or more. 205

7-23 Distribution of people's daily interactions with conversational agents. 1/3 of people do not interact with them, and most people interact at least once with them. 206

7-24 Distribution of participants reported monthly performance of tasks. We see that most tasks have been done rarely, except for organizing folders and sending emails with images. 207

7-25 Distribution of the procedures that were assigned to participants. Once more, the selection process follows an inverse of the reported performance, with composing an email and organizing into folders being some of the least performed. 208

7-26 Distribution of whether a participant was able to accomplish the goal they were given or not. We see that most interactions (70%+) were able to accomplish their goals with the given agents. The GPT based agent was able to guide more interactions successfully. 208

7-27 Distribution of whether a participant was able to accomplish the instructions that the agent gives. We see that most people were able to follow the instructions that the agents gave. Interactions with the GPT based agent were followed more successfully. 208

7-28 Failed GPT-based agent conversation. We can see the GPT-based agent making up very broad instructions on setting a signature in "G. email" 210

7-29 Failed GPT-based agent conversation. We can see once more that the GPT agent tries to make very broad instructions on how to make a reminder Shortcut. Eventually, the agent mixes in the information of how to make a general reminder, but it does not mix it well within how to make the automation. 211

7-30 Failed GPT-based agent conversation. We can see the agent always trying to get more information and not starting the process even though the user indicates this. 212

7-31 Distribution of whether an agent gave an appropriate amount of information or not. People thought the GPT based agent gave more appropriate amounts of information and leaned towards disagreeing on the baseline 213

7-32 Distribution of whether a participant found the agent conversational or not. People found the GPT agent conversational in contrast to the baseline agent. 214

7-33 Distribution of whether an agent was easy to use or not. People leaned towards agreeing on the GPT based agent, in contrast to disagreeing on the baseline. 214

7-34 Distribution of whether a participant would use the agent frequently or not. In both cases, people leaned towards not utilizing the agents frequently. 215

7-35 Distribution of whether the conversation felt natural or not. In the case of the GPT based agent, people leaned towards agreeing it was natural, in contrast to disagreeing with the baseline. 215

7-36 Distribution of whether a participant had to use special phrasing or not. Interestingly, people felt that they needed special phrasing for both agents. 216

7-37 Distribution of whether participants would learn to use the agent quickly or not. People leaned towards disagreeing with the baseline agent, and agreeing with the GPT based agent. 216

7-38 Distribution of whether an agent gave relevant information or not. People were neutral to positive that both agents gave relevant information. 217

7-39 Distribution of whether participants thought that the agent understood them and helped them achieve their goals.People thought the GPT based agent understood them more than the baseline. 217

7-40 Distribution of whether a participant thought that an agent was unnecessarily complex or not. People disagreed that both agents were complex. 218

7-41 User distribution of familiarity with conversational agents. We see most participants were mostly familiar with voice agents. 223

7-42 Distribution of participant's perception of general usefulness of conversational agents. We see that most people perceive conversational agents as being moderately useful or more. 223

7-43 Distribution of participants reported interaction quantities with conversational agents. We see that 2/3 of participants use their assistants 1 time or more daily. 224

7-44 Distribution of participants reported monthly performance of tasks. We see once again relatively low frequency of performing all task save sending an email with an image attachment. 224

7-45 Distribution of the procedures that were assigned to participants. . . 225

7-46 Distribution of whether a participant was able to accomplish the goal they were given or not. Most people were able to complete the original task, even if they were not able to recover from an error. 227

7-47 Distribution of whether a participant was able to accomplish the instructions that the agent gives. Most people were able to follow the instructions that the agent gave. 227

7-48 Distribution of whether a participant was able to resolve the error that they were given. More than half of the interactions were able to recover from errors. 227

7-49 Interaction where exception handling intent fails. We can see that there is no clear indicator that an error has occurred. Later on, we see that the person does declare there was an error, but in the form of "[ERROR DESCRIPTION], how do I fix this" 227

7-50 Distribution of whether an agent gave an appropriate amount of information or not. Participants leaned to agreeing that it was an appropriate amount of information. 228

7-51 Distribution of whether a participant found the agent conversational or not. Most people thought that the agents were not conversational. 229

7-52 Distribution of whether an agent was easy to use or not. Most people did not find the agents easy to use. 229

7-53 Distribution of whether a participant would use the agent frequently or not. Most people would not use the agents frequently 230

7-54 Distribution of whether the conversation felt natural or not. Most people did disagree that the conversation was natural. 230

7-55 Distribution of whether a participant had to use special phrasing or not. Most people agreed that they had to use special phrasing with the agents. 231
7-56 Distribution of whether participants would learn to use the agent quickly or not. Most people disagreed that others would learn to use the agents quickly. 231
7-57 Distribution of whether an agent gave relevant information or not. Most people leaned towards that the agents gave relevant information. 232
7-58 Distribution of whether participants thought that the agent understood them and helped them achieve their goals. Most people were neutral to negative that the agents understood them. 232
7-59 Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people were neutral on whether the agents were unnecessarily complex. 233
7-60 User distribution of familiarity with conversational agents. Most people were moderately familiar or more with agents. 235
7-61 Distribution of participant's perception of general usefulness of conversational agents. Most people thought voice agents were moderately useful or more. 235
7-62 Distribution of participants reported interaction quantities with conversational agents. Most people interacted with agents 1 or more times, although almost half did not interact with them regularly. 236
7-63 Distribution of participants reported monthly performance of tasks. We note that in this sample of participants, many had not made a calendar invite with video, and many (as in the other studies) had composed an email with an image attachment. 236
7-64 Distribution of the procedures that were assigned to participants. We note that from the reported procedures, the one that people had done the least is composed the calendar entry with a video link, so our selection strategy picked this one the most. 237

7-65 Distribution of whether a participant was able to accomplish the goal they were given or not. Most interactions were able to complete their goals. 238

7-66 Distribution of whether a participant was able to accomplish the instructions that the agent gives. Most interactions were able to follow the agents' instructions. 238

7-67 Distribution of whether a participant was able to resolve the error that they were given. We note that almost everyone was able to recover from errors (70%+) for both agents, and all but one recovered with the GPT based agent. 238

7-68 Distribution of whether an agent gave an appropriate amount of information or not. We see that participants found that the GPT agent gave more appropriate amounts than the baseline. 239

7-69 Distribution of whether a participant found the agent conversational or not. Most participants were neutral on whether the GPT agent was conversational, and people disagreed that the baseline was conversational. 240

7-70 Distribution of whether an agent was easy to use or not. Most people leaned towards agreeing that the GPT based agent was easy to use in contrast to the baseline which was seen as not easy to use. 240

7-71 Distribution of whether a participant would use the agent frequently or not. Most people were neutral on whether they would use the GPT-based agent in contrast to the baseline which people would not use frequently 241

7-72 Distribution of whether the conversation felt natural or not. Most people felt that the GPT-based conversation was more natural than the baseline conversation. 241

7-73 Distribution of whether a participant had to use special phrasing or not. Most people found that they needed to use special phrasing with the baseline and were neutral with respect to the GPT agent. 242

7-74 Distribution of whether participants would learn to use the agent quickly or not. Most people thought the GPT agent would be quicker to learn to use than the baseline agent. 242

7-75 Distribution of whether an agent gave relevant information or not. More people felt that the GPT agent gave relevant information and were neutral on the baseline agent. 243

7-76 Distribution of whether participants thought that the agent understood them and helped them achieve their goals. Most participants felt that the GPT agent understood them better than the baseline agent. . . . 243

7-77 Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people disagreed that the GPT-based agent was complex, and were neutral on the baseline. 244

7-78 Distribution of whether participant were able to succeed in their goals or not. Most interactions were able to succeed in their goals, but the ones with the knowledge driven hybrid had higher success rates . . . 250

7-79 Distribution of whether participant were able to succeed in what the agent told them to do. We see once more that most interactions were able to succeed in what the agent guided them on, but the knowledge driven hybrid had a higher success rate of participants being able to follow its instructions. 250

7-80 Distribution of whether participant thought the agent's content was accurate. On the baseline ChatGPT agent, people were neutral leaning to agree, but there was a significantly higher agreement that the hybrid approach was more accurate 250

7-81 Distribution of whether participant thought the agent's content was reliable. On the baseline ChatGPT agent, people were neutral leaning to agree, but there was a significantly higher agreement that the hybrid approach was more reliable 251

7-82 Distribution of whether participant thought the agent's content was accurate and reliable. On the baseline ChatGPT agent, people were neutral leaning to agree, but there was a significantly higher agreement that the hybrid approach was both more accurate and reliable 251

7-83 Distribution of whether an agent gave an appropriate amount of information or not. We see that participants found that the GPT agent gave more appropriate amounts than the baseline. 252

7-84 Distribution of whether a participant found the agent conversational or not. Most participants were neutral on whether the GPT agent was conversational, and people disagreed that the baseline was conversational. 252

7-85 Distribution of whether an agent was easy to use or not. Most people agreed that both agents were easy to use. 253

7-86 Distribution of whether a participant would use the agent frequently or not. Most people thought that they would use both agents frequently 253

7-87 Distribution of whether the conversation felt natural or not. Most people felt that both agents were somewhat conversational 254

7-88 Distribution of whether a participant had to pay special attention to phrasing or not. People were neutral on whether special attention phrasing needed to be used, and slightly more agreed that it was needed on the Hybrid system. 254

7-89 Distribution of whether participants would learn to use the agent quickly or not. Most people thought that both agents could be learned to use quickly. 255

7-90 Distribution of whether an agent gave relevant information or not. More people felt that the Hybrid agent gave relevant information than the plain ChatGPT based agent. 255

7-91 Distribution of whether participants thought that the agent understood them and helped them achieve their goals. Most participants felt that both agents understood and helped them achieve their goals. 256

7-92 Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people agreed that both systems were not complex, with slightly more agreeing on the Hybrid system. . . . 256

8-1 LLMs step generation performance in specific domain. People that have played the video game, will know that Shadowmere is a horse, and that horses are not found in a chest. 264

8-2 Bing chat, powered by GPT-4, failing on providing specific instructions for how to get Shadowmere. 265

8-3 LLM-based conversation in which the language model does not begin an extended telling or does not move the conversation forward towards guidance. 268

8-4 Conversation in which a LLM is not given directions on how to interact with users. 269

8-5 Conversation in which a LLM is given directions on how to interact with users. We note that the answers are shorter and to the point, and wait for a user to continue. 270

8-6 Conversation in which a LLM is given directions on how to interact with users. We note that the model does not exactly follow the given instructions, when an error occurs, but it approximates them. 271

8-7 Providing an example conversation as a prompt for a LLM. 272

8-8 Follow up of providing an example conversation as a prompt for a LLM. We see that the LLM hallucinates an entire conversation to a certain correct degree. 273

8-9 Conversation in which a LLM does not have up-to-date information and begins hallucinating when pressed on a topic that it does not have knowledge on. 275

8-10 Outdated, zero-shot, ChatGPT generated procedure for how to share location on an iPhone. 276

8-11 Generated text in which a LLM is given contextual commonsense assertions to help with procedural step generation. 278

8-12 Asking the hybrid KDPSG + LLM model to explain its reasoning for a specific step that the base model does not know how to do. 279

8-13 Asking the hybrid KDPSG + LLM model for an explanation of a known step based on additional KG information 279

9-1 Dall-E's generation of prompt: *Write the date in the line in the upper-right hand of a paper check. Use a wikiHow art style.* We note that while it is a rough image, there is some semblance of a check and of writing something. 283

B-1 Distribution of overall rating of interactions. We see the Google Assistant interaction rated as neutral, while the ChatGPT ones rated positively and the KDPSG ones rated slightly negatively. 369

B-2 Distribution of whether a participant succeeded in following steps given by the agent. We see that the Google Assistant had mixed results with half the people failing to follow what the assistant said, the KDPSG system trailing behind it, and that in most interactions people could follow what the ChatGPT agent said. 369

B-3 Distribution of whether a participant succeeded in accomplishing their goals. We see a similar distribution to people's rating of the interaction. The Google Assistant satisfied half of the participant's goals, the KDPSG system satisfied slightly less than that, and the ChatGPT agent satisfied most people's goals. 369

B-4 Overall interaction rating. We see that the Google Assistant and the KDPSG assistant were rated close to neutral, and that the ChatGPT based agent was rated close to very good. 370

B-5 Distribution of whether an agent gave an appropriate amount of information or not. We see that people were neutral on intent-based agents and agreed on the ChatGPT agent. 371

B-6 Distribution of whether a participant found the agent conversational or not. Most participants agreed that the ChatGPT agent was conversational and disagreed that the intent based ones were. 371

B-7 Distribution of whether an agent was easy to use or not. Most people leaned towards agreeing that all agents were easy to use. 371

B-8 Distribution of whether a participant would use the agent frequently or not. People agreed that they would use the ChatGPT agent frequently and disagreed on the intent-based ones. 371

B-9 Distribution of whether the conversation felt natural or not. We see that the intent-based agents were rated neutral to disagree and that the ChatGPT one was rated higher than agree. 372

B-10 Distribution of whether a participant had to use special phrasing or not. Most people found that they needed to use special phrasing with the intent-based agents and leaned towards disagreeing on the ChatGPT agent. 372

B-11 Distribution of whether participants would learn to use the agent quickly or not. Most people leaned towards agreeing that all the agents could be learned quickly. 372

B-12 Distribution of whether an agent gave relevant information or not. The ChatGPT agent was rated as giving the most relevant information, and the intent based agents people leaned towards disagreeing. 372

B-13 Distribution of whether participants thought that the agent understood them and helped them achieve their goals. We see that the KDPSG agent was rated as the least understanding/useful for goal accomplishment, and the ChatGPT and Google Assistant being rated higher. This once more suggests that the intent mechanism needs to be made robust. 372

B-14 Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people disagreed that the agents were complex. 372

B-15 Distribution of participants that were able to accomplish their goal. We see that most people were able to with both agents. 376

B-16 Distribution of participants that were able to complete the task that the agent gave them guidance. Once more, we see that most people were able to accomplish what the agent suggested doing 376

B-17 Distribution of interactions that were able to recover from errors. We see that with the ChatGPT based agent most people could recover, however with the intent-based agent not many people could recover. In the interactions, and from the usability feedback, we see that this in large part is due to lack of understanding in the intent detection/slot filling 376

B-18 Distribution of whether an agent gave an appropriate amount of information or not. We see that the participants find that both agents give an appropriate amount of information. 377

B-19 Distribution of whether a participant found the agent conversational or not. Most participants agreed that the ChatGPT agent was conversational and were neutral on the KDPSG. 377

B-20 Distribution of whether an agent was easy to use or not. Most people leaned towards agreeing that the ChatGPT agent was easy to use and were neutral, leaning to negative on the KDPSG.. 377

B-21 Distribution of whether a participant would use the agent frequently or not. People agreed that they would use the ChatGPT agent frequently, and leaned towards disagreeing on the KDSPG. 377

B-22 Distribution of whether the conversation felt natural or not. We see that leaned towards disagree for the KDPSG and agree for the ChatGPT agent. 377

B-23 Distribution of whether a participant had to use special phrasing or not. Most people found that they needed to use special phrasing with the intent-based agents and leaned towards disagreeing on the ChatGPT agent. 377

B-24 Distribution of whether participants would learn to use the agent quickly or not. People agreed that the ChatGPT agent would be quick to use and were neutral leaning to positive on the KDPSG. 377

B-25 Distribution of whether an agent gave relevant information or not. The ChatGPT agent was rated as giving the most relevant information, and neutral on the KDPSG. 377

B-26 Distribution of whether participants thought that the agent understood them and helped them achieve their goals. We see that the KDPSG agent was rated neutral leaning to positive, and the ChatGPT based agent was rated as, agree. 378

B-27 Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people disagreed that the agents were complex. 378

C-1 DESIRE Chatbot Interaction (left), DESIRE Intermediate Procedural Representation (right)[56] 382

C-2 Flow graph generated by DESIRE[56] 383

C-3 Dependency visualization for XPAD system [103] 384

C-4 ProStruct decoder architecture that utilizes commonsense constraints to filter out impossible/improbable states [167] 386

C-5 Inferred procedure flow graph from [98]. 388

C-6 Redesign of a recipe as a workflow[106] 389

C-7 Visualization of boundaries of the different categories of knowledge injections. 391

C-8 RetroGAN System Architecture 392

C-9 Dynamic graph construction example for [35]. When the system reads the **bolded** sentence, it adds a leaf node into its knowledge graph and updates the states of other entities in the graph. 394

C-10 Dynamic graph construction and reasoning example for [16]. The authors build hops and perform inference on the combination of these hops to determine what is the likeliest output. 395
C-11 Visualization of instruction tasks for the system presented by [123] . . 395
C-12 Architecture for system described in [123]. The system takes in a pair of instructions and observations of the environment and produces an action. 395
C-13 Procedure utilized by [107] to respond with natural language text, utilizing a knowledge graph as a grounding mechanism to guide the conversation 397
C-14 Architecture for TRAINS-95 system[49] 399
C-15 Architecture for COLLAGEN based-system[150] 400
C-16 Architecture for PACO[153] 400
C-17 STEVE embodied agent (left), architecture for STEVE (right)[153] . 401

List of Tables

1.1 Simple example of how a user would interact with our agent for procedural guidance and error recovery 54

2.1 The three different instructions in the procedure of sending a fax. Examples were adapted from [44] 62

2.2 The ways that each specific type of instruction can be improved/hindered with regards to initial performance/transfer and learning. Adapted from [44] 63

4.1 Example of inputs and outputs for the GLUCOSE trained contextual commonsense inference model with hints. The *hint* is **bolded** and the parts of the *hint* are colored (subject, relation, object). Without a hint we can see that the model tries to infer directly on the content of the sentence, however with *hints*, the model tries to include an inference based on the target sentence with the contents of the *hint*. 92

4.2 Averages of 4 different seeds over 10 epochs for *hinted* (Hint) and non-hinted (No Hint) runs of the ParaCOMET dataset from [50]. The largest scores are **bolded** and significantly different scores have an asterisk (*) next to them. We can see from the results that *hinted* systems tend to achieve higher performance even if slightly and in some cases significantly, and do not decrease performance significantly. For significance we use the t-Test: Paired Two Sample for Means. . . 97

4.3 Averages of 4 different seeds over 5 epochs for *hinted* (Hint) and non-hinted (No Hint) runs of the GLUCOSE contextual inference task dataset. This is the same dataset as the work in [114] The largest scores are **bolded** and significantly different scores have an asterisk (*) next to them. Once more, we see that hinting provides a small, increase in performance, all the while permitting controllability. We use the t-Test: Paired Two Sample for Means for significance 97

4.4 Results of human evaluation of ParaCOMET and GLUCOSE datasets. The largest scores are **bolded** and significantly different scores have an asterisk (*) next to them. We sampled 100 test points for each model from their test datasets and had the hinted and non-hinted models infer assertions. Humans judged these assertions on a 5 point Likert scale where above 3 was plausible similar to [50]. On the left we can see the average values of the human judgments and on the right we can see the percentage of plausible inferences (rated >= 3). We can see that hinting provides comparable performance. 98

4.5 Here we can see the available specificities in ConceptNet, ATOMIC 2020, and GLUCOSE. We mark with ✗ the specificities that are not available by default, and add * to those that can be generated. 102

4.6 Here we see two examples of instantiating general information based on the context of the story of John and his dog to generate story-specific information. 103

4.7 Here we present the results of our joint inference tests. We color code sets of rows as a testing run on ATOMIC 2020, GLUCOSE, and ConceptNet. The Training Set(s) column contains the knowledge bases that were used to train the model. The Test set column contains which knowledge base test set was used to evaluate the models. The Hint column represents the Hints that were given to the model during testing. Overall, we can see that with hinting on the test set (i.e., hinting the subject or the subject and the relation type), the addition of knowledge bases for inference does not improve nor degrade substantially the performance. To view this in the table, we can compare the rows in which the Hint column is either “Subject” or “Subject, Relation, Specificity”. Additionally, we can see that without hinting on the test set (i.e., rows that Hint is “No”), the addition of knowledge bases for inference tends to decrease the performance. 107

4.8 Results for human annotation of 100 randomly sampled assertions from ATOMIC 2020, GLUCOSE, and ConceptNet test sets and the inferred commonsense from these. We have three sets of three columns Acceptable, Contextually Acceptable, and Alignment Acceptable. Each set of columns is color-coded to represent a knowledge base. Firstly, the Acceptable column is the ratio of whether humans thought that inferred assertions, without context, were acceptably commonsense or not. The Contextually Acceptable, column represents the ratio of whether humans thought that inferred assertions given the context, were acceptable or not. Lastly, the Alignment Acceptable column is whether humans thought that the gold standard (from a knowledge base) assertion was correctly matched to the context. We can see that without hinting, the joint inference model (i.e ATOMIC 2020 - ConceptNet - GLUCOSE - No Hint) improves the acceptability, both with and without context, of assertions predicted in the ATOMIC 2020 test set. We can also see that performance does not degrade much in whether it produces assertions that are contextually acceptable throughout the test sets. We can see that with hinting, however, the performance is decreased and becomes closer to what the individually trained models can achieve. This suggests that with hinting, the model tries to channel the knowledge base that we are targetting, and aligns with what we see in the automated metrics. 108

4.9 We present the results of the adversarial test with ablations. We can see that the Adversarial models tend to have improved recall (ROUGE) scores, but lower precision (BLEU/METEOR) scores and lower accuracy on classifying a ConceptNet test set of assertions. The adversarial model with everything (+ADVERSARIAL+CONFOUNDER) strikes a balance of the benefits of precision and recall that the non-adversarial, and non-confounder loss models give respectively. 116

4.10 Table that describes extracted fields from the scraper. 120

4.11 Fields and their descriptions for the index that stores contextual procedure facts. 121

5.1 Validation and test results for training models for procedural step generation with and without knowledge. 136
5.2 Results for test and validation averages for varying amounts of contextual commonsense assertions. We see the trend that generally, the more assertions given, the better the performance. Around 64, we see that performance starts to become unsteady. This is most likely due to context length constraints, since we are using the FlanT5 model, the context is limited to 1024 tokens. 140
5.3 Results on the validation and testing set for different ordering styles. We see that the rouge-based filtering provides the best performance, but note that to get this score, we need to know the step we are going to generate ahead of time. Whereas random ordering gives the worst performance, and plausibility-based ordering gives an intermediate performance. 141
5.4 Results of complete (with the procedures that have a generated knowledge graph) training, and random sampling of knowledge with rouge filtering. 142
5.5 Here we see the actual values for the metrics used to evaluate whether the approximation for the rouge score for filtering would be effective. 144

6.1 Example of extended telling pattern (A3.0) taken from [108] 159
6.2 Hearing Check (B1.1.0) pattern taken from [108] 160
6.3 Paraphrase Request (Agent) (B1.1.0) pattern taken from [108] 160
6.4 Repeat (User) B2.1.0 pattern taken from [108] 161
6.5 Paraphrase Request (User) B2.4.0 pattern taken from [108] 161
6.6 Inquiry (User) A1.0 pattern taken from [108] 162
6.7 Confirmation A5.2 pattern taken from [108] 162
6.8 Disconfirmation A5.3 pattern taken from [108]. 163

6.9 Completion Check A5.4 pattern taken from [108] 163
6.10 Open Request A2.0 pattern taken from [108] 163
6.11 Opening Self-Identification C1.1 pattern, taken from [108], and applied to our conversational agent 164
6.12 Modified extended telling pattern. This occurs in the case that a simple inquiry triggers a longer explanation of how to do a process. 170

7.1 Side-by-side example of golden instructions vs. procedural instructions. We note that this procedure was unseen by the knowledge driven system. 195

Chapter 1

Introduction

> *Start by doing what's necessary; then do what's possible; and suddenly you are doing the impossible. - St. Francis of Assisi*

1.1 Preamble

In recent years, machine learning has been used to perform astonishing tasks, such as autonomous driving vehicles [158], self-landing rockets [175], and even systems that teach themselves to play video games [172]. Although we have progressed far and wide in applying machine learning to a variety of tasks, the seemingly simple task of providing automated instructions with error-handling capabilities seems to have been evaded by researchers. Now, with the advent of transformer-based natural language processing systems [14], it has suddenly become a very real possibility to develop a fully automated voice-based procedural guidance system. In this work, we will describe one way to achieve this.

1.2 Motivation

Instructed learning (for example, how-tos) is commonplace throughout our daily lives (e.g., "How do I prepare rice?" or "How do I send images to my family from my iPhone?", and many more). We routinely learn or teach new procedures through sets of instructions communicated through conversation or written documents. Alternatively, one can obtain direction for procedures by viewing examples, improvising, conversing with other people, or receiving instruction from a skilled individual. It has been found that human beings find reading instructions intellectually taxing and would often prefer to be guided through a procedure by another person [44].

Modern conversational agents, such as Alexa[1], Siri [2], or Google Assistant[3], offer a scalable platform that could be leveraged to try and help users in instructed learning. However, only a very small subset of procedures (e.g., cooking [45]) are implemented and can be told by the agent. Additionally, these procedures are implemented by scraping popular sites verbatim for the written instructions, which in turn can produce some awkward steps which in the website were intended to be additional information. Since starting this work, more general how-to's have now been made accessible in Alexa using the wikiHow skill [6]. However these "conversational" skills, provide very limited capabilities: forwards, backwards, repetition, and a predetermined set of procedures available for guidance. This makes them relatively inflexible for users, especially in the case that errors occur in the procedure, or alternatives are needed in steps.

It is worth noting that procedure-related "How-to" questions were once the second most prevalent type of query in web queries [37]. We could only guess that for modern conversational agents such as Siri, Alexa, or the Google Assistant, that have a user-base in the millions, and that presumably people ask questions from all parts of life, the number of "how-to" queries is just maybe even more voluminous. This indicates a large demand in this area for a way to satisfy this need for guidance. This is further demonstrated by the large amount of instructional how-to content found in online video sites such as YouTube. Given all of this, it is incredibly frustrating that most modern conversational agents tend to give results similar to those seen in Figures 1-1 and 1-2. We see that they either give a small snippet of what might be the procedure

---

[1] https://en.wikipedia.org/wiki/Amazon_Alexa

[2] https://www.apple.com/siri/

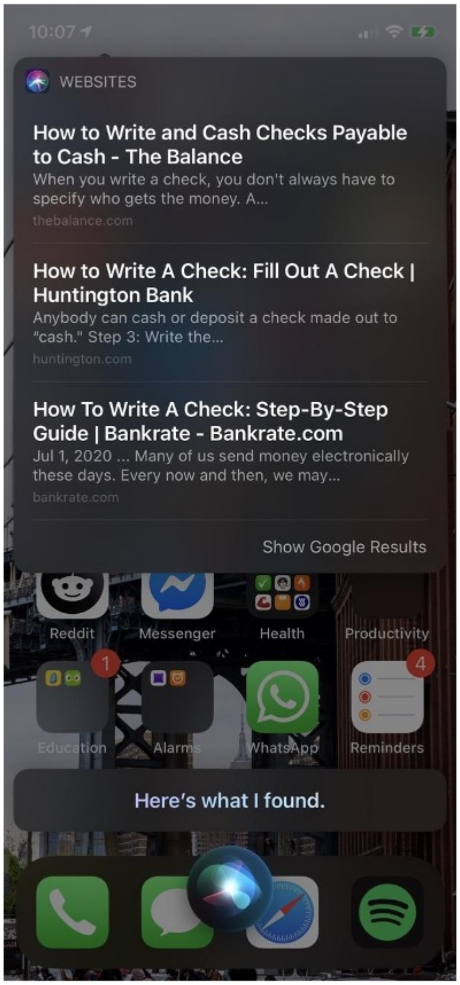
[3] https://assistant.google.com

[4] https://support.google.com/googlenest/answer/7309433?hl=en

[5] https://www.amazon.com/alexa-kitchen-hacks-recipes/b?node=21440412011

[6] https://vixenlabs.co/learn-how-to-do-anything-with-the-wikihow-alexa-skill

or answer, or a list of web results. In the case that the assistants return a list of web search results, this has been seen as a failure on the agent's part [96].



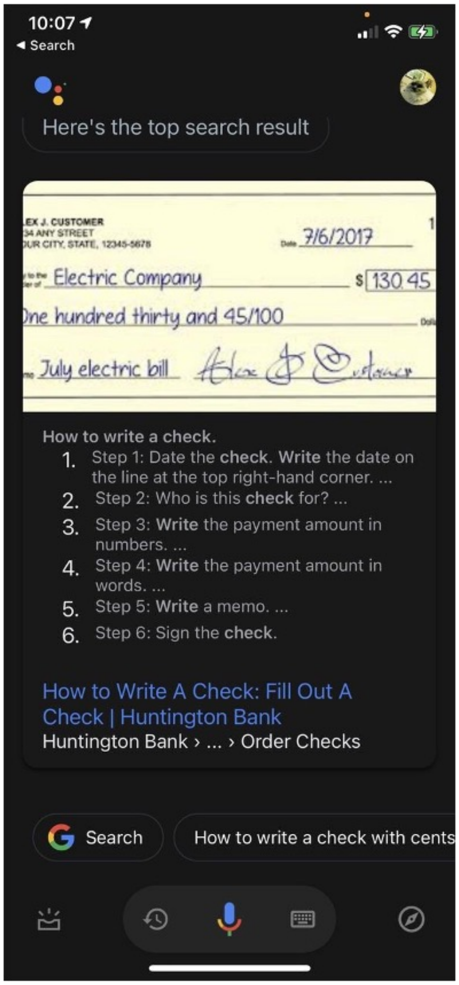
How to Write and Cash Checks Payable to Cash - The Balance
When you write a check, you don't always have to specify who gets the money. A...
thebalance.com

How to Write A Check: Fill Out A Check | Huntington Bank
Anybody can cash or deposit a check made out to "cash." Step 3: Write the...
huntington.com

How To Write A Check: Step-By-Step Guide | Bankrate - Bankrate.com
Jul 1, 2020 ... Many of us send money electronically these days. Every now and then, we may...
bankrate.com

Here's what I found.

Figure 1-1: Behavior of Siri when asked for guidance . The assistant falls back to a web search for the topic and gives no guidance.



Here's the top search result

How to write a check.
1. Step 1: Date the **check**. **Write** the date on the line at the top right-hand corner. ...
2. Step 2: Who is this **check** for? ...
3. Step 3: **Write** the payment amount in numbers. ...
4. Step 4: **Write** the payment amount in words. ...
5. Step 5: **Write** a memo. ...
6. Step 6: Sign the **check**.

How to Write A Check: Fill Out A Check | Huntington Bank
Huntington Bank › ... › Order Checks

Figure 1-2: Behavior of Google Assistant when asked for guidance. The assistant gives a copied and pasted response from Wikihow as guidance.

More recently, very large language models, with human-like responses have been made publicly accessible through chat interfaces. A popular example of these is ChatGPT[7]. These large language models have enough parameters that they are capable of memorizing some procedural knowledge and can provide access to it through chat-like prompting. However, although these models seem to produce incredible results, they are still capable of hallucinating knowledge [69]. One alternative to minimize this, is

---

[7] https://openai.com/blog/chatgpt/

to train/fine-tune these very large language models on a specific domain, however that may be time consuming and expensive. One thing that is also lacking for voice-driven guidance in these very large language models, is that they are trained on written text, which may not necessarily translate effectively to an actual spoken conversation and may not follow patterns of how we, as humans, communicate.

Prior to this, there has been research into automated systems for guiding people through procedures. We discuss these in detail in the *Background* chapter and *Prior Work* subsections later on. In summary, these works have focused on two distinct aspects of procedural interactions: *how procedural information is presented* and *how procedural information is understood*. What this means is that some projects do not consider how to obtain the knowledge and instead assume it is provided, focusing exclusively on the communication component. On the other hand, the other set of systems does not consider how to convey effectively the information that they have. Instead, they focus on novel techniques of extracting and reasoning with the information. This creates a dichotomy without any connecting work between good procedural understanding techniques and good procedural communication techniques.

To the best of our knowledge, no attempt has been made to develop ***a conversational agent with an underlying end-to-end system capable of automatically understanding, abstracting, and generating on-the-fly procedural instructions***. In our work, we present a novel knowledge driven procedural step guidance system (KDPSG). We name this agent for user interactions as Winston. A conversational agent that can automatically abstract, understand and generate on-the-fly procedural guidance. A system like this would be beneficial to society, since it would enable conversational agents to teach users new practices and procedures interactively via a conversation on a device, rather than deferring to a web search or providing a highly constrained interaction. Additionally, the work relies on contextual procedure knowledge graphs, which may enable conversational agents to engage in explainable, informed conversational experiences related to procedures, similar to the system in [107], which uses a KG to discuss a particular topic.

This intermediate contextual KG also allows modifications to the knowledge to address issues such as mistakes, misunderstandings, and alternate methods of performing a procedure. Additionally, the work serves as a starting point for understanding procedures for other means such as linking textual directions to actions that can be performed on a smartphone. With the use of the intermediate knowledge graphs, we could learn effective mappings between sets of assertions and a corresponding Siri shortcut action for example. If such a technique were applied it could teach unskilled users how to operate a piece of technology and could even enable end-user voice programming. Altogether, with this work, we move towards bridging the gap between modern conversational agents and modern procedural understanding systems.

1.3 Overview

In the following subsection, we give a very broad overview of the different techniques and systems that we have developed to be able to create a conversational agent for procedures.

1.3.1 Abstracting procedures to knowledge graphs

To have a machine understand a procedure and be capable of reasoning through it to produce steps, the procedure needs to be abstracted into a contextual Knowledge Graph (KG) and operated on by a separate reasoner. We define a KG as a collection of knowledge, in the form of tuples, that represents information that is true about the world. We give a more formal definition later in Chapter 2. This abstraction of a procedure into a KG can enable things such as combining several methods of completing the procedure (i.e., many sources) and removing/modifying information in the graph to accommodate changing circumstances (e.g., a person is on a diet and wants a substitute for white rice in a cooking procedure). Additionally, it may enable systems to discover various paths through the graph that might be used to complete the same procedure in a different way.

Briefly, to perform this abstraction of a procedure into a knowledge graph, we utilize a contextual commonsense inference model which we describe in Chapter 4.

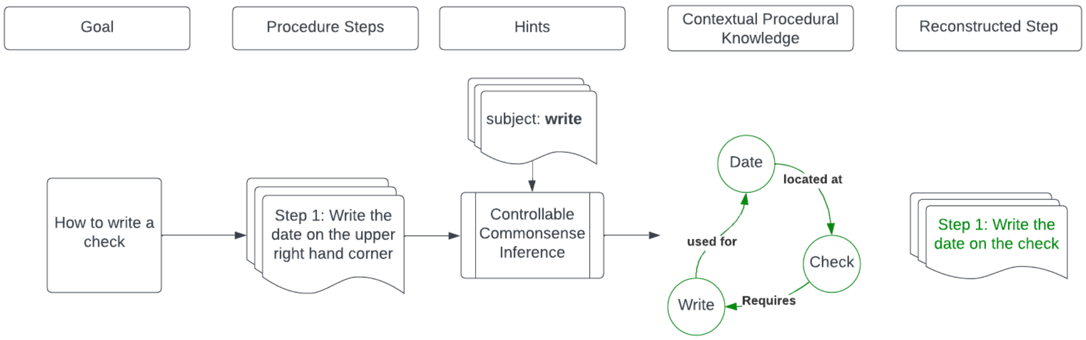

Goal
Procedure Steps
Hints
Contextual Procedural Knowledge
Reconstructed Step
subject: write
How to write a check
Step 1: Write the date on the upper right hand corner
Controllable Commonsense Inference
Date
located at
used for
Check
Write
Requires
Step 1: Write the date on the check

Figure 1-3: Overview of the process for generating contextual procedural knowledge graphs and steps. From left to right, a given text (e.g., story or procedure description) is iterated sentence by sentence. For each sentence, we provide a hint of what a contextual commonsense inference model should infer a fact about. We then collect these facts for a contextual knowledge graph of the input text.

What this model does is that it takes in a certain textual context (in our case a procedure) and goes sentence-by-sentence inferring facts that may stem from the sentence within the specified context. The model is also controllable and can infer facts along specified dimensions of commonsense and about entities/events. Once the model has iterated through all the sentences using the controlling mechanism which we call "hinting" to derive facts about entities/events in the sentence/context, we are left with a collection of facts that can be viewed as a contextual knowledge graph of a procedure which represents all the knowledge necessary to accomplish or understand a certain procedure. A visualization of this process can be seen in Figure 1-3. We give much more details of this process of controllable commonsense inference in Chapter 4.

1.3.2 Reasoning over procedural knowledge graphs for step generation

Once we have abstracted the procedures into knowledge graphs, we proceed to reason on these graphs to generate steps. To do this, we develop a ranking system that goes through the knowledge available for a procedure and ranks it based on the plausibility of the assertion given the context. Although this ranking is not perfect, it provides good performance and can be done as assertions are generated. We leave for future work, exploration of more complex, reinforcement learning methods for assertion plan generation. With this ranking, we pick the top 64 facts and pass them into a graph-to-text system that generates the next procedural step. By operating in the knowledge space, it is also possible to combine different knowledge graphs to be able to resolve issues that may occur within procedures. With the contextual knowledge graphs, we have a prior of all the information available up to the moment the failure occurred. We can then combine this knowledge of how to perform an error-handling procedure and the prior knowledge of what was happening on the step, to be able to generate steps for sub-procedures or sub-goals that would solve a person's issue and put them back on track to complete their original goal.

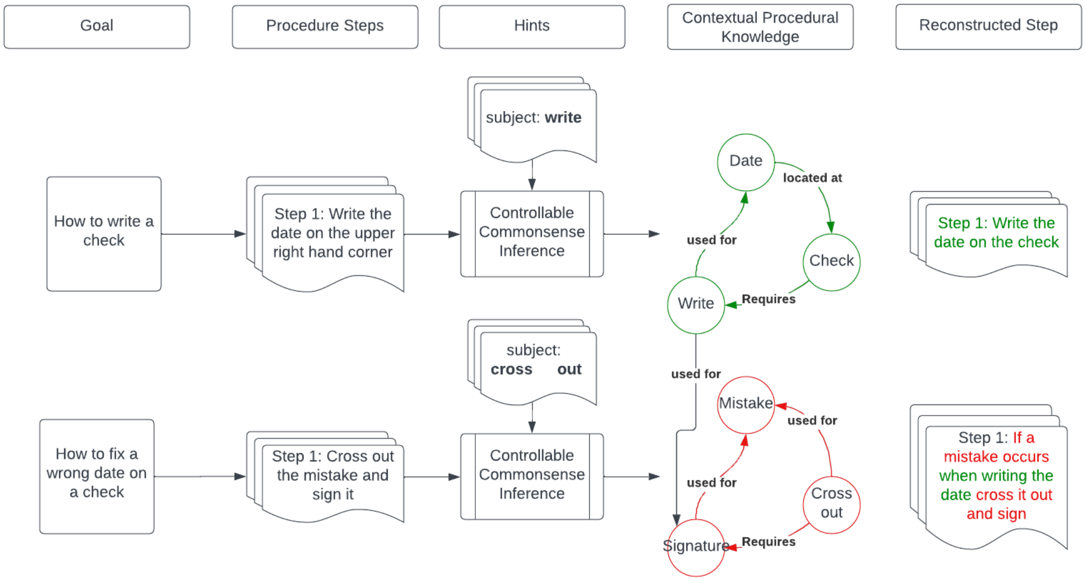
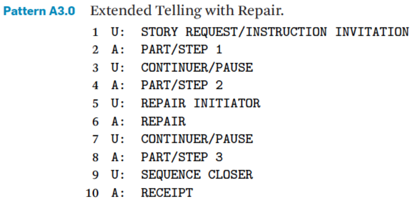

Goal
Procedure Steps
Hints
Contextual Procedural Knowledge
Reconstructed Step
subject: **write**
How to write a check
Step 1: Write the date on the upper right hand corner
Controllable Commonsense Inference
Date
located at
used for
Check
Write
Requires
Step 1: Write the date on the check
subject: **cross out**
used for
Mistake
used for
How to fix a wrong date on a check
Step 1: Cross out the mistake and sign it
Controllable Commonsense Inference
used for
Cross out
Signature
Requires
Step 1: If a mistake occurs when writing the date cross it out and sign

Figure 1-4: Overview of the process for generating contextual procedural knowledge graphs for error handling. Similarly to the previous figure, the controllable commonsense inference model is used to generate the contextual knowledge graph of the procedure to perform error handling. However, to generate a step for the procedure, the knowledge graph from the prior goal is utilized in addition to the one for the error.

1.3.3 Conversational agent and patterns for communicating steps

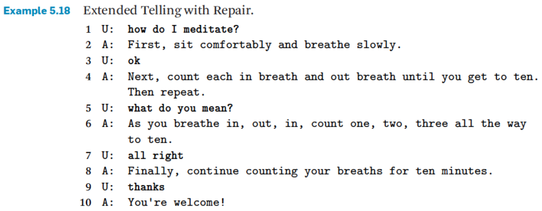
Finally, now that we have generated a contextual knowledge graph from a procedure and leveraged it to generate steps for a procedure, we have to provide these to people so that they can utilize them to actually complete a procedure. To accomplish this, we developed a conversational-pattern-based agent. These agents provide a natural way for users to receive guidance. Additionally, conversational agents can provide users with the ability to ask follow-up questions, making it easier for users to understand the agent's response.



Pattern A3.0 Extended Telling with Repair.

```
 1  U:  STORY REQUEST/INSTRUCTION INVITATION
 2  A:  PART/STEP 1
 3  U:  CONTINUER/PAUSE
 4  A:  PART/STEP 2
 5  U:  REPAIR INITIATOR
 6  A:  REPAIR
 7  U:  CONTINUER/PAUSE
 8  A:  PART/STEP 3
 9  U:  SEQUENCE CLOSER
10  A:  RECEIPT
```

Figure 1-5: Extended telling conversational pattern. Taken from [108]



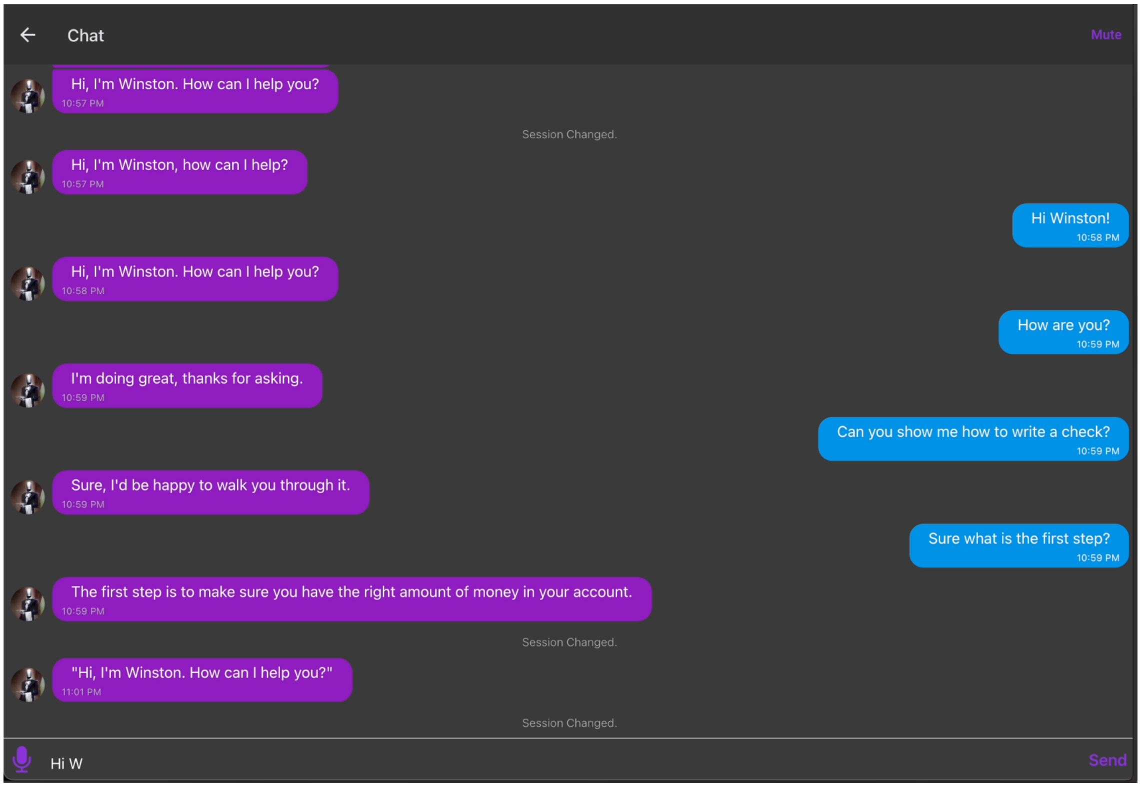
Example 5.18 Extended Telling with Repair.

```
 1  U:  how do I meditate?
 2  A:  First, sit comfortably and breathe slowly.
 3  U:  ok
 4  A:  Next, count each in breath and out breath until you get to ten.
        Then repeat.
 5  U:  what do you mean?
 6  A:  As you breathe in, out, in, count one, two, three all the way
        to ten.
 7  U:  all right
 8  A:  Finally, continue counting your breaths for ten minutes.
 9  U:  thanks
10  A:  You're welcome!
```

Figure 1-6: Extended telling example. Taken from [108]

To develop a conversational agent we used conversational patterns [108] that have been developed in an effort to handle interactions in a manner that the end user feels conversational. These patterns have been extracted from watching many conversations between individuals from the field of Conversational Analysis [108]. These patterns serve as a template that when supplied with the correct information, and with an adequate combination of patterns can dictate the behavior of an agent that actually does conversational procedural guidance.

In our particular work, we make heavy usage of the Extended Telling [108] pattern, which can be seen along with an example in Figures 1-5 and 1-6. In addition to this, we leverage other patterns such as the Open Request [108] to be able to handle users' requests for guidance. We give a detailed explanation of the patterns and how they are arranged for interactions in Chapter 6. Additionally, we implement these patterns utilizing IBM's Watson assistant in combination with a custom Flask server that supplies required information (e.g., steps). To make the system conversational rather than a chatbot, we incorporate speech-to-text and text-to-speech capabilities within a React Native interface that has the conversation transcript. We can see this interface in Figure 1-7.



Hi, I'm Winston. How can I help you?
10:57 PM

Session Changed.

Hi, I'm Winston, how can I help?
10:57 PM

Hi Winston!
10:58 PM

Hi, I'm Winston. How can I help you?
10:58 PM

How are you?
10:59 PM

I'm doing great, thanks for asking.
10:59 PM

Can you show me how to write a check?
10:59 PM

Sure, I'd be happy to walk you through it.
10:59 PM

Sure what is the first step?
10:59 PM

The first step is to make sure you have the right amount of money in your account.
10:59 PM

Session Changed.

"Hi, I'm Winston. How can I help you?"
11:01 PM

Session Changed.

Hi W

Figure 1-7: Example of Graphical User Interface (GUI) utilized to communicate with our conversational agent. The outputs of the agent are relayed to users through speech-to-text, and the user communicates with the agent by pressing the microphone button and recording a message.

1.3.4 Guiding Example

To take all of this together, we now give an example of how the system would operate. We frame the scenario such that it is a person that wants to get guidance with an agent on their phone. The particular task that the person wants guidance with, is “how to write a check”.

We set up a hypothetical conversation with an agent in Table 1.1. In this conversation, we can see firstly that a user requests guidance for writing a check. Then we see that the user commits a mistake, and notifies the mistake to the agent. The agent then begins guidance on how to fix the mistake. We see how the user progresses through the conversation until they finish.

Actor	**Example**
User	How do I write a check?
Agent	The procedure I know how to do is writing a check, is this OK?
User	Yes exactly
Agent	Alright! The first thing that you need to do is date the check.
User	Where do I write the date?
Agent	On the line in the top right corner
User	I made a mistake! I wrote the wrong date.
Agent	No worries! Let's try to fix this. Cross out the mistake and write the correct date.
User	What's next?
Agent	Sign next to the crossed out mistake.
User	Ok
Agent	That's it!
User	Ok then! What do I do now?
Agent	The next step in writing a check is to write the name of the recipient
User	What's next?
Agent	Write the amount in both word and number forms.
User	Anything else?
Agent	Yes, the last step is to sign the check!
User	Got it thanks!
Agent	You're welcome!

Table 1.1: Simple example of how a user would interact with our agent for procedural guidance and error recovery

Breaking this interaction down, we see that there was a request for guidance, a question about the procedure, a request for error recovery within the procedure, then continued guidance on the original procedure. We see that the procedural guidance follows the extended-telling conversational pattern, in which the larger procedure is broken into steps, one at a time. We also see that the error recovery follows the same pattern after declaration of the error. Ultimately, the person finishes the procedure and gives thanks. Once more, all of this is handled by conversational patterns. These can be instantiated by using an intent-driven system.

Now, we look into the first instruction that was given: “The first thing that you need to do is date the check”. If we are using a knowledge graph to generate this content, we would need facts such as:

- Step has the property first
- Step requires writing
- Writing is used for the date
- Writing requires pen
- The date is located in the upper-right-hand corner
- Check contains date

With these facts, we could translate them into a sentence such as the one we see in the Table. Now, with this knowledge graph at hand, when the user makes a mistake, such as writing the wrong date, the system can take into consideration that a pen was used, and that the mistake needs to be crossed out. Altogether, we could come up with a set of contextual facts that can be translated to this error handling. In our work, we present a contextual commonsense inference system that is capable of generating such facts. We also present a realizer system that can take these facts and convert them into a procedural step such as the ones that we see in the example. Now, these generated facts can be given to a conversational management system that implements the conversational patterns that we mention before to bring about an experience such as the one given in our example.

1.4 Research Questions and Novelty

1.4.1 Research questions

In this work, we address the following research questions to build an automated agent with modern natural language processing systems that can provide guidance in procedures.

- Can all the necessary knowledge of a procedure be abstracted successfully into a knowledge graph?
    - Importance and Relevance:

      If a procedure *cannot* be abstracted into a knowledge graph (with the appropriate context for the knowledge) there may be no effective way to reconstruct the procedure or find alternative ways of performing it. Furthermore, operating within the KG space to add additional information on how to perform error resolution would be extremely limited or impossible, since error-resolution procedures cannot be abstracted successfully either.

- Can I utilize a contextual knowledge graph of a procedure to generate conversational steps for a procedure?
    - Importance and Relevance:

      This question goes hand-in-hand with the prior one. If we have a successful abstraction mechanism, we need a successful reconstruction mechanism. With such a mechanism, we can then generate a variety of ways to complete a procedure. Additionally, this would suggest that operations that we perform in the knowledge graph space (such as removing or adding certain facts to handle an error or personalize a procedure) could be converted back into the language space.

- Can I combine procedure knowledge graphs to handle exceptions in a procedure?
    - Importance and Relevance:

      In this question hinges the error-handling capabilities of our system. If we cannot combine knowledge graphs effectively, or at least a way to leverage new information when changing a goal within a procedure, we have no way of being able to use the information to be able to guide users out of an error.

- Can I use the generated steps in natural conversational patterns to effectively guide a human through a procedure and the exceptions that may occur in it?
  - Importance and Relevance:

    Although conversational patterns should produce natural/conversational interactions when utilized in agents, we are evaluating them with information that is dynamic (e.g., generated steps) in addition to a situation (e.g., procedure) that may fail at any point. We need to make sure that the patterns can still function and that a user can complete a procedure whenever this happens.

- Can I establish some guidelines for the development of conversational agents that handle procedures?
  - Importance and Relevance:

    Finally, we are developing a system that is conversational, but it is limited in its understanding of the world; it is only capable of reading/processing textual content. Because of this, it is very likely that certain types of procedures, such as intricate assemblies, or procedures that require good visual descriptions, may not be effective. By developing and testing our system, we are attempting to give some guidance on which procedures are effective/ineffective within the context of a conversational assistant.

1.4.2 Novelty

Current Limitations

Currently, conversational agents have a very limited capacity to provide procedural guidance in a comprehensive and accurate manner. In the best of cases, instructions from a pre-made set of procedures are available in a limited navigation system (wikiHow skill[8]). In the worst and arguably the most average cases, the agent de-

---

[8] https://vixenlabs.co/learn-how-to-do-anything-with-the-wikihow-alexa-skill

faults to a web search. As a result, there is a need for the development of advanced conversational agents that can provide effective procedural guidance.

Additionally, large language models such as GPT-3 [20] or even ChatGPT [9] could be utilized, out of the box, for this process. Although it is possible, it is important to note that even advanced language models such as GPT-3 and ChatGPT are capable of hallucinating knowledge, which means that they may generate incorrect procedures. This is because these models are trained on vast amounts of text data, including information that may be inaccurate or outdated. As such, other methods, such as the ones that we present in this work, may be needed to give factual accuracy for these large language models in very specific, or constantly changing domains.

Unique aspects of this work

To our knowledge, this work is unique in various ways. First, this is the first research work to use large language model-based systems for procedural guidance tasks within a conversational agent context. Second, this is the first work that utilizes a contextual commonsense inference system to perform procedural understanding, along with a ranking strategy to filter procedural contextual knowledge. Third, this work is the first work that leverages conversational patterns with dynamically generated information for open-domain instructional dialogue. Finally, this is the first work to utilize contextual commonsense knowledge on-demand as an aid for very large-language model guidance systems to update their knowledge on unseen/outdated procedures.

Implications

The completion of our work leaves the following implications. First, contextual knowledge graphs can be utilized for the generation of procedural steps. This means that any improvements in filtering, ordering, and reasoning of contextual knowledge graphs, can be leveraged to produce more useful steps, and in the future may be used to provide alternative ways of performing a procedure. Second, conversational

---

[9] https://openai.com/blog/chatgpt/

patterns with dynamically generated information, can help improve the overall user experience, providing more natural and engaging conversations. Overall, this work has the potential to make task-oriented conversational agents more effective and efficient in providing procedural guidance. The system presented in this work provides an alternative for handling the static response of "here is a web search that may have your answer" with a more concrete answer, at least in the case of procedural guidance.

1.5 Contributions

We now provide a list which summarizes all the contributions of this work.

1. **Controllable Contextual Commonsense Inference System**

   A novel transformer-based, sequence-to system that is capable of taking a story, a sentence from that story, and a hint, and controllably produces a contextual commonsense assertion (fact) that is relevant to the story and sentence. The system is novel in that it utilizes the hint to guide the generation process, it uses a co-trained classifier to rate the plausibility of the assertions, and a single contextual commonsense inference model learns commonsense information from multiple aligned sources.

2. **Graph-to-Step System & Assertion Ranking Strategy**

   A novel transformer-based sequence-to-sequence system that takes a procedure name, a number of a step, prior steps, and a set of assertions and produces the indicated step for a procedure. This is the first system which applies the Data-to-text task for procedural step generation. This is combined with grounding the contextual inference process and sorting assertions by plausibility to select the essential knowledge for a procedural step.

3. **Conversational Procedure Error Handling Strategy**

   Novel utilization of an expanded Inquiry conversational pattern to handle procedural errors. Inquiries for error handling procedures are treated as Extended

Tellings to provide error handling guidance. In addition to this, we combine contextual knowledge graphs to contextualize the generated steps for this error handling extended telling.

4. **Conversational agents for procedural guidance & error handling**

   Design and implementation of conversational agents that provide guidance with our novel contextual commonsense system. We show that the use of this conversational agent with the patterns along with our dynamic step generation system can be utilized to guide people through procedures and through error recovery in procedures.

5. **Hybrid Agent that uses Large-Language Models in combination with Graph-to-Step and Contextual Commonsense Inference**

   Implementation and evaluation of a novel conversational agent that leverages the flexibility of large language models and the explainability and knowledge-based approach of the contextual commonsense and graph-to-step system. We show that this hybrid system produces more accurate and reliable information that a plain large-language model. We also provide a first evaluation of large-language models applied to general procedure guidance

6. **Guidelines for Future Agent Development**

   From our evaluations and feedback we provide a list of guidelines that should be taken into consideration when developing conversational agents for procedures.

Chapter 2

Background

> *When I was young, I had to learn the fundamentals of basketball. You can have all the physical ability in the world, but you still have to know the fundamentals. - Michael Jordan*

2.1 Procedures and Instructions

Procedural inquiries in the form of "How-To" questions used to be the second-largest class of questions in web queries [38]. This raises the question of what exactly is a procedure, and how could we provide guidance for this effectively. A procedure is performing a series of tasks to achieve a goal [145]. The knowledge necessary to complete these tasks is known as procedural knowledge [116]. An instance of procedural knowledge is procedural instructions or as they are more commonly known: instructions [44]. In a procedure, there are various ways that instructions can be constructed: the regular *procedural instruction*, (task-oriented instructions), *principles* (system-oriented instructions), or *examples* (instance-oriented instructions). *Procedural instructions* describe how to complete tasks in a step-wise manner, *principles* describe rules governing the tasks, and *examples* demonstrate how instances of the task are carried out [44]. An example of the three types of instructions can be seen in Table 2.1.

Type of instruction	**Example Usage**
Procedural Instruction	To send a fax: 1) Place the document in the tray, text facing up 2) Dial 011 3) Dial the country and the city code
Principle	To send a fax: 1) Place the document in the tray, text facing up 2) Dial 011, then input international access code. This code is used to dial out of countries. The International Telecommunication Union recommends using the standard of 00. 3) Dial the country and the city code
Example	To send a fax to Berlin from a particular fax machine in Iceland: 1) Place document in the tray on the right side of the machine, text facing up. 2) Dial 00-49-30-5555 3) Press the start (green) button

Table 2.1: The three different instructions in the procedure of sending a fax. Examples were adapted from [44]

The content and type of instructions influence their effectiveness for the goals of good initial performance, learning, and transfer. If the goal is good initial performance, then instructions should highly resemble the task at hand (e.g., in the form of detailed procedural instructions and examples), but if the goal is good learning and transfer, then instructions should be more abstract, inducing learners to expend the necessary cognitive effort for learning [44]. These types of instructions can be modified in such ways that they can benefit/hinder both initial performance and transfer and learning of a process. Some modifications and their effects can be seen in table 2.2. All of this taken together means that instructions have to be carefully thought out to be effective for a task at hand. It also implies that good instruction generation needs to have at its disposal a good understanding of why a certain task must be done (Principle), along with concrete, contextual information on how to perform that task (Procedural Instruction, Example). It may also imply that the styling of the instructions needs to be controllable to give more or less detail or to give examples.

Type of instruction	Initial Performance		Learning and Transfer	
	Helped By	**Hurt By**	**Helped By**	**Hurt By**
Procedural Instruction	•Being specific in the step descriptions •Combining a general step description with a principle •Combining a general step description with a specific example •Giving a specific goal with the instruction	•General step description	•General step description •Combining a general step description with a principle •Exercise with goal provided	•Being specific in the step descriptions •Giving a specific goal with the instruction
Principle	•Combining with a general step description		•Combining a general step description with a principle •Adding principles that clearly relate to the interface at hand	•Adding principles to specific procedural instructions •Providing principles only
Example	•Similar to task at hand	•Less similar to task at hand	•Emphasizing Sub-goals	•Using examples without support to guide generalization

Table 2.2: The ways that each specific type of instruction can be improved/hindered with regards to initial performance/transfer and learning. Adapted from [44]

2.2 Knowledge Graphs

2.2.1 General Knowledge Graphs

In this work, we define a knowledge graph (KG) as “a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities" [59].

Broadly speaking, KGs are a collection of tuples that represent facts or assertions[1] that should be true within the knowledge of a represented world. As an example, consider the assertion "generally, a dog is an animal". In our work, we represent this assertion as the tuple: *(general, dog, is a, animal)*, where each element of the tuple (demarcated by a comma “,”) is: *assertion type*, *subject*, *relation*, and *object* respectively. The *assertion type* indicates whether the assertion is a *generic* rule or a *specific* instance of a rule. The *subject*, *relation*, and *object* elements are like those found in subject-verb-object triples; however, the relation element may be more than

---

[1] In this work, we use interchangeably “fact” and “assertion”. It is worth noting that a fact is something that is objectively true, whereas an assertion is a declaration of something.

just a verb and subjects and objects need not be noun phrases (e.g., they can be events).

A knowledge graph can be a directed graph, and the edges in the graphs may have weights. In our case, the graph is directed, and the weights of the edges of the graph represent the strength of the assertions. We try to normalize this weight between 0 and 1, but other works may have weights that are not in that range (e.g., ConceptNet [162]) An aspect that makes knowledge graphs useful is that new knowledge can be added and linked or removed and unlinked. Some definitions of knowledge graphs categorize them as a more dynamic system that acquires and integrates information and applies a reasoner to derive new knowledge [43].

Mathematically, a knowledge graph is a tuple $G := (V, E, L)$ [59]. The set of edges (E) or assertions is composed of triples $E \subseteq V \times L \times V$ which are seen as a subject (a concept or event), a relation (a label), and object (another concept or event) respectively (e.g. $(subject, relation, object)$). We extend this definition to additionally include the *assertion type* which can be seen as $A \in \{general, specific\}$; when put together, the knowledge graphs that we work with are in the form of $G := (A, V, E, L)$.

2.2.2 Commonsense Knowledge Graphs

ConceptNet [92] is an example of a commonsense knowledge graph. It is a collection of crowdsourced assertions of what people think is commonsense information. This collection is mostly taxonomic (e.g. a dog is an animal), and not causal (e.g. x causes y). An example of a sub-graph of ConceptNet can be found in figure 2-1. To address this lack of causal information, ATOMIC [155] was built. ATOMIC is another knowledge graph that collects causal, person-related if-then assertions that complement ConceptNet. An example of a sub-graph of ATOMIC can be seen in figure 2-2. Although ATOMIC provides a good causal knowledge graph, it is person-centric and non-contextual which leads to conflating information about an event that may have occurred under different scenarios [113]. To address these shortcomings and shortcomings, also found in ConceptNet, GLUCOSE [113] was collected. GLUCOSE

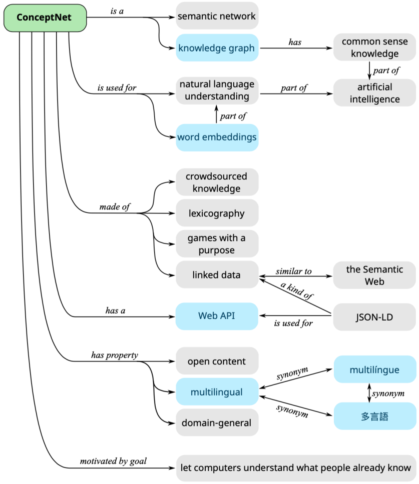


Figure 2-1: Sample sub-graph from ConceptNet [92]

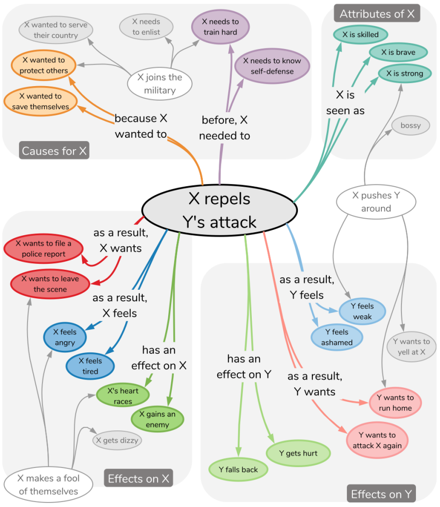


Figure 2-2: Sample sub-graph from ATOMIC [155]

is a dataset that combines assertions with textual examples of them. Although GLUCOSE is a dataset for pairing causes with context, the assertions found in GLUCOSE can be compiled to form a knowledge graph. An example of a subset of data from GLUCOSE can be seen in figure 2-3.

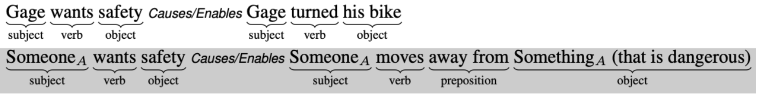

Gage wants safety Causes/Enables Gage turned his bike
subject
verb
object
subject
verb
object
Someone_A wants safety Causes/Enables Someone_A moves away from Something_A (that is dangerous)
subject
verb
object
subject
verb
preposition
object

Figure 2-3: Sample data from GLUCOSE[113]. The sentence in the gray box is the assertion. We can convert these assertions similar to how ATOMIC converts its assertions as seen in figure 2-2

A follow-up work developed ATOMIC-2020 [65]. ATOMIC-2020, which we utilize in our work, is an improved version of ATOMIC that seeks to incorporate some of the taxonomic information found in ConceptNet and expand on the causal information provided by ATOMIC.We use ATOMIC-2020 in our work because it was one of the most comprehensive, documented and well-backed resource available at the time. However, we do note that the ATOMIC work has expanded more recently to distilling knowledge from very large language models such as GPT-3 [178]. The resulting KG from this work we leave as future work to explore contextualizing and incorporating into our work.

Lastly, we incorporate Ascent++ [119] into our work. ASCENT++ was developed through automatic construction methods, extracting the information from the C4 corpus [142]. Some of the salient features, apart from the scale of the data used for the automatic construction, are that it has refined expressiveness and both better precision and recall than prior works [119]. It captures composite concepts with subgroups and aspects, and by refining assertions with semantic facets. This is useful because it helps express the temporal and spatial validity of assertions and further qualifiers [119]. This work can be seen as a massive addition to both ATOMIC and ConceptNet and contains approximately 2.4M assertions.

2.3 Transformers and Natural Language Processing

In this work, we rely heavily on the utilization of Transformer-based systems [171]. A transformer-based system is a neural model that relies on the use of an attention mechanism [8, 171] to build powerful and generalizable contextual representations of an input text. Through the attention mechanism, tokens in an input text learn to "attend" to other tokens in the input text to build a representation for themselves. The original system developed in [171] was an encoder-decoder model utilized for sequence-to-sequence tasks. We give a simplified overview of this model in Figure 2-4. In this model a transformer-based encoder would generate a hidden representation of a text, and a decoder would convert that hidden representation into some other output. Around the same time that the transformer architecture was introduced, studies were being carried out to determine how best to train models for effective transfer learning in the language domain [62]. Following this, it was discovered that if these systems were given large amounts of data, they could be utilized as pre-trained models, and through transfer learning and fine-tuning, could be adapted to a wide variety of tasks [40, 140]. Two of the most popular systems used to achieve state-of-the-art scores in a wide variety of tasks were BERT [40] and GPT-2 [141]. Briefly, BERT is a transformer-based model that consists of *only* the encoder part from the original transformer work. To be able to utilize only this encoder model for downstream

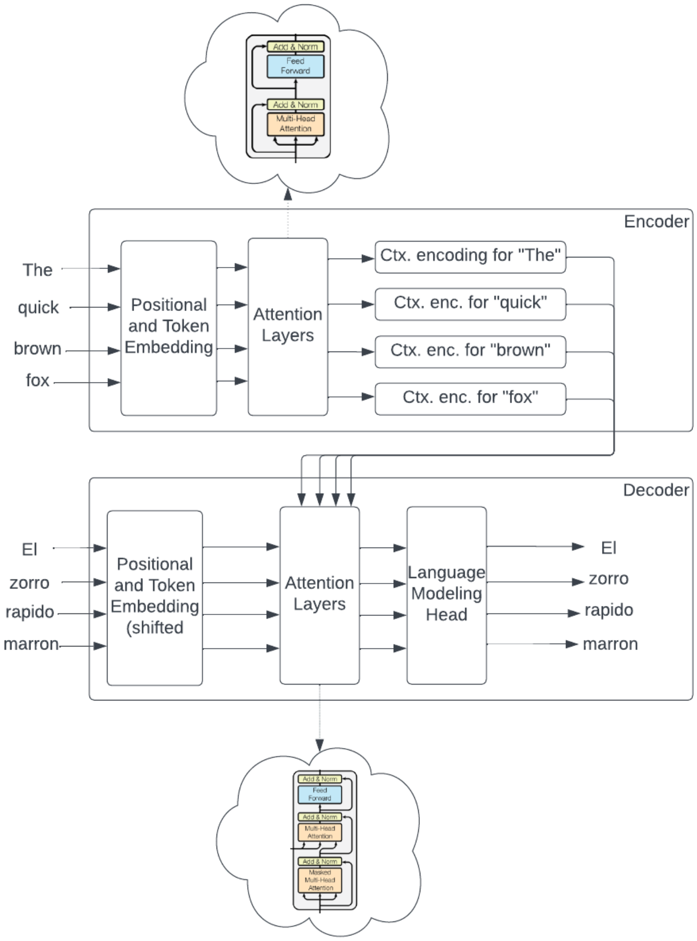


Figure 2-4: Simplified overview of the encoder-decoder transformer model. Cutouts and overall architecture from [171]

tasks, the authors developed a masked language modeling task. This consisted of a Cloze-like approach [169] (i.e., fill in the blanks) in which the model had to figure out a masked token, from the other tokens surrounding it. Some issues that arose from this were that it was not immediately clear how this could be utilized to generate language, as there was no apparent way for the model to recurrently generate text. Later, this was addressed [174], but by then encoder-decoder architectures were being reintroduced as being capable of doing this and performing better in other downstream applications. GPT-2 or the General Pretrained Transformer 2, is a transformer-based model that consists of *only* the decoder part of the original transformer-based model.

This model was trained on the regular language modeling objective (which consists of predicting the next token in a sentence, given all the prior ones). The way this was done utilizing this transformer model, which had no sense of recurrence, was by masking out future tokens when performing the self-attention. In other words, the model can only see the past and current token, when predicting the next token.

Following the success and wide application of these models to a variety of tasks, it was then discovered that different strategies could be employed when pre-training these transformer models that could yield improved performance [94, 184]. It was also found that by scaling these systems, both in the amount of data used in pretraining and in parameters, the models could achieve even higher performance when used for downstream tasks [142, 20, 79]. Additionally, it was found that different pre-training strategies could be utilized in the originally proposed encoder-decoder transformer model, and achieve comparable and in many cases better performance on downstream tasks than any of the encoder-only or decoder-only approaches [79, 142]. In addition to this, these encoder-decoder models could function as general sequence-to-sequence models that had already been applied to many tasks, such as translation [8].

More recently, there have been some promising directions in this field of large-language models. One of the more interesting of those is the incorporation of external knowledge from sources such as knowledge graphs. We have explored some of the ways of injecting this knowledge into transformer encoder-based models, such as BERT [30]. In general, there appeared to be many promising methods to incorporate structured information. One of the most popular, and simple, was utilizing additional information (e.g., facts) as additional prompts for a language model. We explored some of this work to examine the incorporation of information through graph embeddings and found that there was indeed an improvement in performance in downstream tasks [186]. Another area that has been promising is that of prompting [93]. Prompting is a series of techniques in which a prompt in the form of text, or a vector, is strategically chosen or learned to tap into information that a language model may have acquired from pre-training. We give more details of this in Chapter 3.1.1. Finally, as of this writing, one of the more recent and promising directions has been to utilize reinforcement learning with very large language models to have them refine their outputs based on feedback on certain tasks [27]. This type of refinement has been shown to increase general task performance and zero-shot capabilities of large-language models. In our work, we utilize these models, (in particular the FLAN-T5 model) as they are the current state-of-the-art in various benchmarks.

2.3.1 Very Large Language Models

Recently, it has been seen that there is a strong correlation between increasing the amount of parameters in a transformer-based model (i.e., transformer layers and context size), and the performance of such models. Large language models, typically larger than 10B parameters, such as GPT-3 [20], ChatGPT[2], and LLaMA [170] and others have shown state-of-the-art performance, in a zero-shot setting in many NLP tasks. Many people have also started researching what is called emergent behaviors in language models. This is the phenomenon that after a certain amount of parameters and training, a very large, transformer-based language model can begin to perform reasoning-like behavior in some tasks[177]. Some researchers have even begun to speculate that the dawn of artificial general intelligence has come about with extremely large, multi-modal, language models [21]. While only time will tell, these models permit extremely flexible behaviors like filtering information, extracting information, and even serving as chatbots. However, given these model's scales, and that they are based on the transformer architecture, these models can still hallucinate or come up with misinformation [68]. Given that these large language models are very good zero-shot systems, yet are expensive to train and fine-tune, retrieval based systems are an excellent way to supply these models with accurate information that they may need to improve their zero-shot reasoning. We do note, that there is a very large body of work on utilizing reinforcement learning to fine-tune and train these large-language models to minimize some of the hallucinations that they have and to improve zero-shot reasoning capabilities [157, 95]. Later on in Chapter 6, we will show how we can leverage our contextual commonsense inference work to supple-

---

[2] https://chat.openai.com/

ment large-language-models. We also look at utilizing these large-language models as a proxy for a reasoner system.

Chapter 3

Related Work

If I have seen further, it is by standing on the shoulders of Giants -Isaac Newton

3.1 Relevant work for Abstracting Procedures into Facts

3.1.1 Large Language Model Prompting

Recently, there has been a shift in Natural Language Processing from pre-training and fine-tuning a model to pre-training, prompting, and predicting [93]. One reason for this shift is the creation of ever-larger language models, which have become computationally expensive to fine-tune. Prompting is finding a way to convert a model's input sequence into another sequence that resembles what the model has seen during pre-training, which makes it recall certain information. In our work, we consider the task of prompting for language generation, an open-ended formulation.

Recall that prefix prompting modifies the input to a language model, by adding either a hard prompt (additional words to the input sequence) [160] or a soft prompt (i.e., adding trainable vectors that represent, but are not equivalent to, additional words) [86, 78, 93]. Unlike classic prefix prompting, the technique we develop and present in the next chapter,*hinting*, uses both hard and soft prompts. The soft prompts are in the form of symbols that represent the different parts of the assertion (i.e., subject (<*subj*>), relation type (<*relation*>), and an object (<*obj*>)), and the hard prompts are in the form of the actual parts of the assertion that are selected to be appended as part of the hint as seen in our example in section 3.1.1. Hinting is similar to KnowPrompt [24], except that they use a masked language model and soft prompts for relationship extraction. AutoPrompt [160] is also similar, but finds a set of "trigger" words that give the best performance on a *cloze*-related task, whereas we provide specific structured input for the model to guide text generation.

We classify hinting as a hybrid prefix-prompting technique due to the inclusion of trainable symbols that are not part of the model's original vocabulary and can be viewed as soft-prompts given to the model, as well as the combination of these soft-prompts with actual hard prompts to generate a contextual inference. We refer to [93]'s definition of prompting as a three-step process. To begin, we define a function that converts the input to an intermediate template and fills it out with information about the target task (i.e. contextual commonsense inference. In our case, by including what we call the hint, a random subset of the tuple that is to be predicted, between parenthesis at the end of the input (i.e., the input of the model follows the structure of [STORY CONTEXT]*(HINT)*). Second, the model must generate an answer using this template, hence the prefix prompting. Finally, as training progresses, the prompt is directly mapped to a Y, which corresponds to the target contextual assertion in the contextual commonsense inference task. We give much more details on the actual process for hinting later in Section 4.3.1.

3.1.2 Controllable Generation

Controllable generation can be described as ways to control a language model's text generation given some kind of guidance. One work that tries to implement controllable generation is CTRL [74]. The authors supply control signals during the pre-training of a language model. This approach is intended to provide a generally applicable language model. A body of work in controllable generation has focused on how it can be used for summarization. Representative work that uses techniques similar to ours is GSum [42]. In contrast to GSum, our method of hinting is model-independent, allows for the source document to interact with the guidance signal, and contains soft prompts in the form of trainable embeddings that represent the parts of a tuple. The GSum system gives an interesting insight into the fact that highlighted sentences, and the provision of triples, does in fact help with the factual correctness of abstractive summarization.

We make the distinction that hinting falls more under prompting for the reason that we utilize additionally the trainable soft embeddings rather than purely additional hard tokens and that our task of contextual commonsense generation is not explored in the controllable generation works, whose main focus is on controlling unstructured text generation. Some works that are in this area are also [127] who utilize what they call ”control factors” as keywords or phrases that are supplied by a human-in-the-loop to guide a conversation. More similar to our work, but tailored for the task of interactive story generation and without trainable soft-embeddings, is the work by [18] which uses automatically extracted keywords to generate a story. In future work, we could possibly utilize the automatic keyword extraction to supply parts of a hint, rather than our approach of complete parts of an assertion, and expand this to utilize synonyms and antonyms of keywords. During inference time, later for procedural understanding, we do utilize keyword extraction along with noun-phrase and verb-phrase extraction to produce procedural contextual commonsense facts. Additionally, we are exploring synonymy-based hinting and have found that it does significantly help with the recall ability of the model, however, we do not have any conclusive results yet. Lastly, there is the work by [159] which looks at controllable text generation for the purpose of conversation and utilizes an embedding to give quantitative control signals as part of conditional training.

3.1.3 Story and Assertion Alignment

The closest works to ours, with respect to constructing a story-aligned assertion dataset, are ParaCOMET and GLUCOSE [50, 114]. GLUCOSE uses human annotation to perform the alignment between stories and commonsense assertions. Para-

COMET takes an automated approach in which assertions are aligned either by giving the sentence to a COMET model as an input and producing a relevant inferred assertion or by calculating the cross entropy of combining the story up until the target sentence with an assertion from a knowledge base. Our method differs from this in that we utilize the cosine distance between semantic representations of the story and its sentences and an assertion from a knowledge base. Some possible difference that arises from this is that our method could match assertions that may not be explicit in a story to that story. Whereas ParaCOMET's approaches, which are based on cross-entropy for coherence and n-gram matching, are likely to produce assertions that have parts that are explicit in the text. Overall, our approach can match more abstract assertions to stories. Additionally, our method permits us to use the optimized FAISS library to scale up to billions of stories and assertions, and gives us the freedom to select how to embed the stories/sentences/assertions.

More recently, the work by Gao, et.al., (ComFact [51]), does a similar procedure to how we do the joint alignment. They perform contextual alignment with parts of the triple, utilizing the same cosine distance approach. It is worth noting that apart from the ROCStories dataset, they also utilize a conversational dataset for additional contextual fact matching. This work came out later as we were conducting our studies, but it provides good insight into some of the problems that the task of contextual fact-matching systems must face.

3.2 Relevant work for Processing Abstracted Procedural Knowledge into Steps

3.2.1 Text Generation and Reasoning with Knowledge Graphs

Although knowledge graphs are useful for reasoning about questions and procedures, they can be used for other purposes such as text generation and guidance for the text generation. Some of the aforementioned works generate a reasoning chain to arrive at an answer, but in our case, we need to convert this reasoning chain to a natural language instruction.

The line of work by Moryossef et al. [110, 109] addresses this by breaking up neural text generation into a planning system that takes sub-graphs combined with a neural text generation system that generates text based on these sub-graphs. The planning system generates a tree-like structure, which when input into the text generation system produces text that matches the relations/facts in the tree-like structure. This is particularly interesting, because it makes it feasible to go straight from a graph structure into natural language text. If we contextualize this into a procedural system, the reasoner that we would need for procedural step generation could generate a path that represents the steps, and a system such as the one presented by Moryossef could be utilized to translate the path that the reasoner determines is a step, as an input to a text generation system that produces the actual steps that would be relayed to the user.

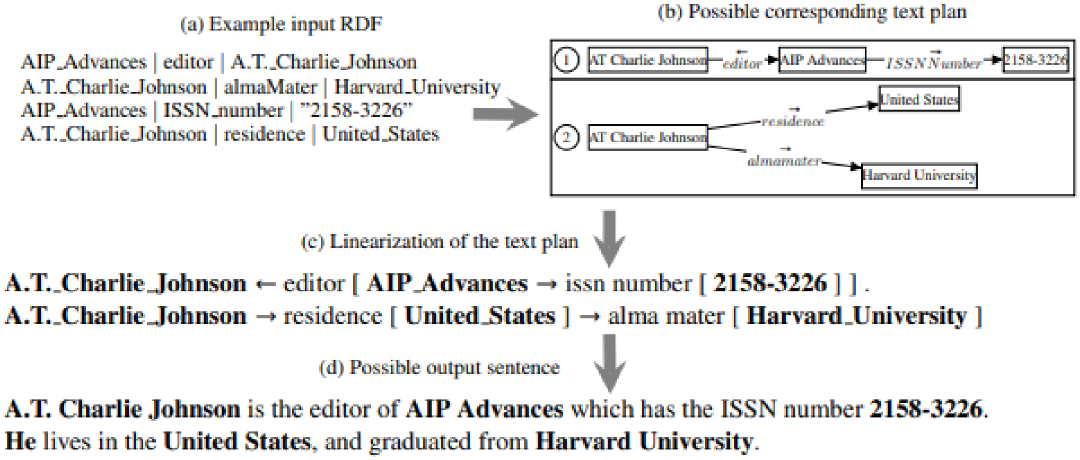

(a) Example input RDF
AIP_Advances | editor | A.T._Charlie_Johnson
A.T._Charlie_Johnson | almaMater | Harvard_University
AIP_Advances | ISSN_number | "2158-3226"
A.T._Charlie_Johnson | residence | United_States
(b) Possible corresponding text plan
1
AT Charlie Johnson
editor
AIP Advances
ISSN Number
2158-3226
2
AT Charlie Johnson
residence
United States
almamater
Harvard University
(c) Linearization of the text plan
A.T._Charlie_Johnson ← editor [AIP_Advances → issn number [2158-3226]] .
A.T._Charlie_Johnson → residence [United_States] → alma mater [Harvard_University]
(d) Possible output sentence
A.T. Charlie Johnson is the editor of AIP Advances which has the ISSN number 2158-3226.
He lives in the United States, and graduated from Harvard University.

Figure 3-1: Procedure utilized by [110] to translate a an input graph into natural language text

There has been some notable work in planning for text generation, apart from the work from Moryossef et al. [110, 109], there has been work at looking at reinforcement learning for automatically generating plans. This began before the explicit need for planning by works such as [135], that provides a way of utilizing reinforcement learning for controllable story generation. Other works have focused explicitly on planning out what the text generation should generate. There has been research into justifying even further the need for planning for data to text generation [45], saying that there is a correlation between higher quality planning and more natural, cohesive text. In our work later on, we show similar results. This work consists of what are micro or sentence-level plans. A similar work, by Puduppully et al., [137], focuses also on this micro planning scale. They train a model that can jointly learn to select content and decode it into text. They find improvements in the number of relevant facts contained in the output text, and the order according to which these are presented. They also see improvements in the grammaticality, and conciseness of the generated text. Although that model seems very useful and the approach well justified, it may not be able to generalize well as it is trained to generate text within a certain domain. In our case, we would like to have a transformer based model such that it could be fine-tuned and perform well even in an out of domain situation. In another work, they consider higher level planning (i.e., planning paragraph and composition level structure) [138]. They find that this scale of planning, is more advantageous for generation tasks expected to produce longer texts with multiple discourse units. However, they note that Other approaches focusing on micro planning (Puduppully et al., [137]; Moryossef et al., [110]) might be better tailored for generating shorter texts. In our work, we are interested in mostly shorter texts (as our data is on average around 10 sentences).

We also note that there have been other works in sentence-level planning [100]. We note that there have been creative approaches to planning for language generation, such as the DYPLOC [63] in which authors use language models to generate plausible alternatives by encoding an input and have a final selector that picks one of them. We also look at some work which shows some promise in pre-training models for graph-to-text [73] by incorporating explicit structure-based graph representations, and at work that looks at using language models for zero-shot graph-to-text generation [83] and shows promise of the abilities of large language models for this task. There is also work that focuses on the controllability aspects of plan then generate systems [165] particularly for table-to-text [84] generations.

More recently, there has been work on utilizing large language models as zero-shot planners for embodied reasoning [64]. This direction seems very promising, particularly because one could specify a task without much complexity (through plain text prompting) and parse through the answer as needed. We look at the work by Yang et al. [183] which is the most applicable work to ours. They develop a reinforcement learning-based planner that can learn to plan the knowledge needed (ordering and selection) for text realization. Their method is particularly interesting because the planner can, based on the text that has been generated or realized until the moment, adapt its planning. In our case, this might be particularly useful if it could navigate through all the available knowledge and find the knowledge needed for an exception. In our work, we try to implement a system similar to the one they propose.

A similar area to this is prompting for language models. Prompting, as explained in Chapter 4, consists of selecting or providing a language model with an input that can help maximize its performance (ideally without any fine-tuning of the model). A work that is of particular interest is RLPrompt [39], in which the authors learn through reinforcement learning and feedback signals, an optimal input for language models for a task. They provide some key insights, such as that the rewards need to be normalized because of the wide range of responses that can be had from the language model. There has also been work on reinforcement learning for guiding text generation [185] and [117], but it is on using RL for learning to generate open-domain text, however it uses semantic similarity for guidance.

3.3 Relevant work for Relaying Steps through Conversational Agents

3.3.1 Modern Collaborative Agents

Modern collaborative agents have started to explore the possibility of automatically mining procedures and presenting them. This somewhat addresses the issue of acquiring domain knowledge, but there is no explicit understanding of underlying procedural knowledge. It is worth noting that there is still a part of the research that is focused on the presentation of the information, however, this seems to have centered around utilizing augmented reality (AR).

Enabling Interactive Answering of Procedural Questions [99]

The work by [99] presents an alternate way of generating intermediate representations (tree-like structures) from procedural instructions. They do this by first identifying the procedure from sources, splitting it into steps, then extracting decision points to generate procedure branches, and finally generate closed domain questions on the decision points to ask a user where to go. The authors do not incorporate it into an actual agent, but they mention that it is the machinery necessary to be able to do procedure guidance. The authors also do not mention what happens if a user makes a mistake in a procedure.

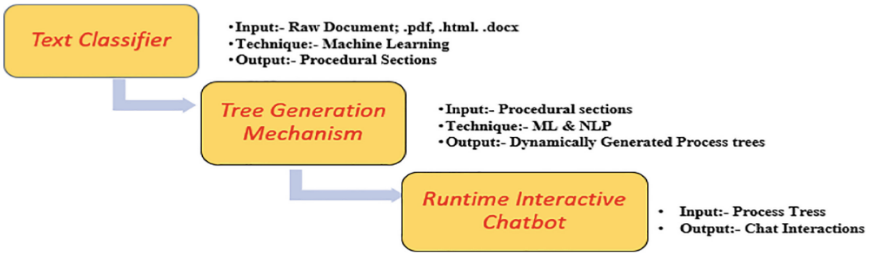

Text Classifier
•Input:- Raw Document; .pdf, .html. .docx
•Technique:- Machine Learning
•Output:- Procedural Sections
Tree Generation Mechanism
•Input:- Procedural sections
•Technique:- ML & NLP
•Output:- Dynamically Generated Process trees
Runtime Interactive Chatbot
• Input:- Process Tress
• Output:- Chat Interactions

Figure 3-2: Architecture for “Enabling Interactive Answering of Procedural Questions" [99]

Cognition-based interactive Augmented Reality Assembly Guidance System (CARAGS) [176]

The other line of more recent work is on how to present the information in a procedure. The work in [176] presents an augmented reality system called CARAGS, which based on the user's cognitive state either presents the information, visually (images, videos, or text) or through audio. For the procedures that the system provides guidance, a domain expert must generate an Ontology for Assembly Tasks Procedure (OATP) to describe the properties and parts of the procedure. The system, based on the cognitive state (perception, attention, memory, and execution), will provide information through a different modality. Once more, this system is primarily concerned with which modality to present the information in, and it is successful in minimizing a user's attention shifts because of this; however, it makes no attempt to automatically generate the OATP to automate the creation of the procedures.

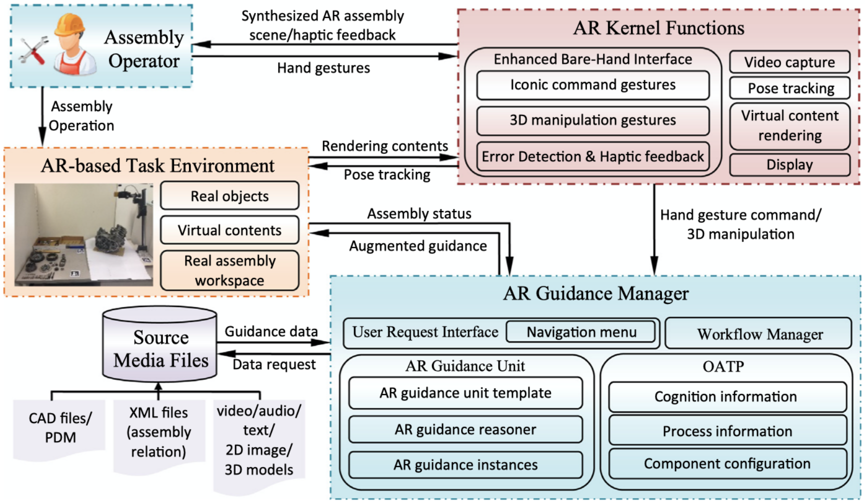


Figure 3-3: Architecture for CARAGS [176]

Intelligent Augmented Reality Training for Motherboard Assembly[179]

The work by [179] focused on incorporating an intelligent tutoring system (ITS) into a mixed reality system for teaching a user how to assemble a motherboard and its components. The authors found that it was extremely useful to incorporate the ITS to help users perform the task. The system that the authors utilize is ASPIRE[105] and based on a description and constraints of the task that is wanted to be performed, it generates a tutoring system to guide users. They combine this with a head-mounted augmented device that provides audio and visual text instructions of the step along with graphics for guidance. This system still suffers from the need to supply all of the domain knowledge necessary to be able to successfully complete the task.

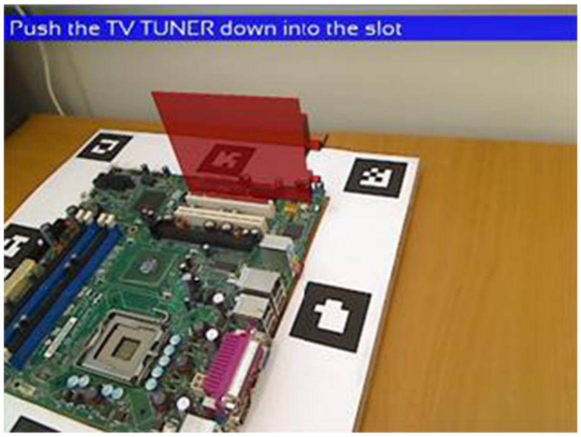


Figure 3-4: Example of MAT agent guidance from [179]

Cognitive Embodied Conversational Agents in Virtual Learning Environment [118]

The work [118] dives back into Virtual Reality and focuses on instantiating an environment with different embodied conversational agents (ECA) imbued with cognitive systems to be able to effectively teach a task. The authors extend the MASCARET [22] to incorporate a Belief Desire Intent-like framework that should be able to give an ECA the ability to convey the correct action based on the world that it is in, to most effectively teach a task. Although they automate the instantiation of the environment and the agents, one must supply a UML description of all the knowledge necessary to carry out the task that will be performed.

Popular Assistants

Currently, some of the most popular, and widely accessible assistants are the Google Assistant, Siri, and Alexa. Siri as of the time of this writing is incapable of giving guidance on procedural tasks. It defaults to a list of web results. The Google Assistant is sometimes slightly better, in that it sometimes gives snippets, directly cut from a website, on how to perform short tasks (around 7 steps); however it only reads out loud the result and has no conversational management skills. It is however capable of guiding users through cooking procedures that once more are scraped from the web. Alexa is also capable of similar cooking guidance, but does not offer snippet guidance

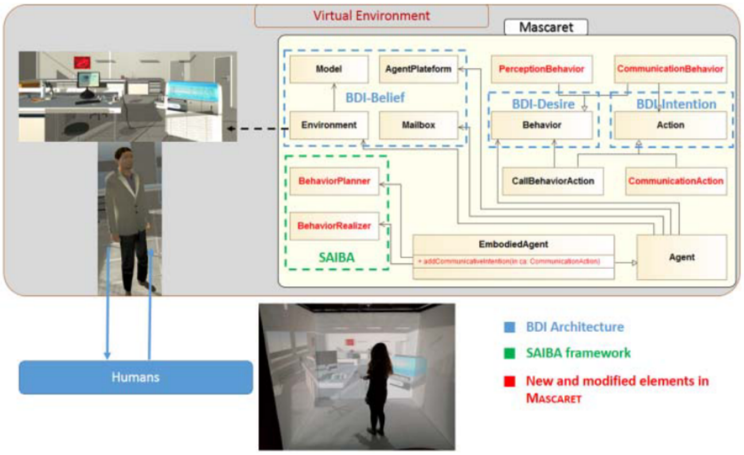


Figure 3-5: Architecture for [118]

nor web search results. Alexa has access to the wikiHow skill, which is the closest thing to an open-domain guidance system.

wikiHow conversational system

The wikiHow skill for Amazon Alexa provides step-by-step instructions on how to do a wide variety of tasks. It has over 180,000 articles that cover a wide range of topics such as cooking, tech, fitness, relationships, life skills and more. You can ask Alexa to open the wikiHow skill and then ask for instructions on how to do something. The skill will then provide you with detailed instructions on how to complete the task. You can also ask Alexa to read the instructions out loud or send them to your phone via text message.

We note that users seem to struggle with similar problems of intent matching and slot filling and that there are some issues in the navigation. A review from the skill's website[1] is the following:

> "You really have to use the exact words that you're looking for to find any answers. It's very frustrating if you'd like to hear something again because you can't make it go back 30 seconds or it'll just start over from the very beginning introducing wikiHow. It's difficult to navigate and

---

[1] The website is `https://www.amazon.com/wikiHow/dp/B01NAI70T7/ref=sr_1_1?keywords=wikihow&qid=1684170734&s=digital-skills&sr=1-1`

> often I'll say next, to go to the next step and it will just go back to the beginning introducing wikiHow again. I wish that there was someway to rewind or go back 15 seconds without it flipping out by Unk canceling everything you've gone through already."

To compare against this, we implement a very similar system, which we call the Gold Instructions later on. It consists of an intent-based system that can take a request for a procedure and look up the instructions from the wikiHow dataset that we scrape.

GRILLBot

The most recent work that is similar to ours is GRILLBot[53]. GRILLBot is the winner of the first Alexa TaskBot Challenge, in which participants have to build a system that can guide people through tasks. GRILLBot is a multi-modal task-oriented voice assistant that guides users through complex real-world tasks for the Alexa TaskBot Challenge. It helps search over a large task corpus with mixed-initiative and executing those tasks with neural dialogue management, knowledge grounded question answering, and web information extraction. To represent each task, TaskGraphs are proposed as a dynamic graph unifying steps, requirements, and curated domain knowledge enabling detailed contextual explanations and adaptable task execution. Automatic linking of multi-modal elements helps the user navigate through the task and enrich the experience with helpful videos and images. Broad use of neural language models makes for flexible chit-chat, contextual intent parsing, and accurate task retrieval. This work was released late 2022 and we were not aware of it until at the end of our work. GRILLBot is similar to the agent we develop, but it differs in that they utilize what they call TaskGraph: "a novel dynamic representation of steps, requirements, and extra elements to walk users through intricate tasks. TaskGraph enables fluid and adaptable conversations by providing a structured abstraction of the task. It allows GRILLBot to take the initiative, or for the user to make decisions on the task at hand.". We note that this is a similar approach to the one we are proposing on generating a contextual knowledge graph, but it does not take into account facts, rather curated content. The authors mention also that it is somewhat heavily curated in the tasks that it can perform. In future work it would be promising to see how this system performs on the tasks that we evaluated, and whether it could be augmented with the contextual knowledge graph that we utilize.

Chapter 4

Abstracting Procedures into Facts

> *Remember that every science is based upon an abstraction. An abstraction is taking a point of view or looking at things under a certain aspect or from a particular angle. All sciences are differentiated by their abstraction. - Fulton J. Sheen*

4.1 Overview

Since the agent that we propose in this work requires knowledge graphs to work, we first look into how we can produce these graphs. These knowledge graphs have to be contextual to a given story, and have to be commonsense, so information that is implied in the process can be made explicit in the graphs. For this, we develop a controllable contextual commonsense inference model that works to generate assertions about a certain context. Then, we examine how we can leverage this model towards our task of converting procedural texts to contextual procedural knowledge graphs.

4.2 Contextual Knowledge Graph Generation

Contextualized or discourse-aware commonsense inference [50] is a task in which we are given a text context (e.g., story) and a selected sentence from that context, and we have to infer a coherent and contextual commonsense assertion (i.e., fact) from the given context and target sentence. We make a note that in our work, a story can be a procedure, but not every story is a procedure. We may interchangeably use story later on to refer to procedures, but it is worth making this distinction. We extend this definition to additionally include story-specific assertion inferences (i.e., templates that are instanced by elements from a story), or general assertion inferences (i.e., fact templates) as used in [114]. This very broad framing of just requiring a textual context and a target sentence is important because the text could be a story or a procedure among others. We give an example of the task in Figure 4-1.

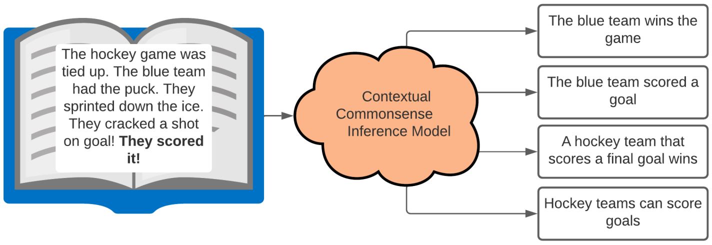

The hockey game was tied up. The blue team had the puck. They sprinted down the ice. They cracked a shot on goal! **They scored it!**
Contextual Commonsense Inference Model
The blue team wins the game
The blue team scored a goal
A hockey team that scores a final goal wins
Hockey teams can score goals

Figure 4-1: Overview of the task of contextual commonsense inference. From the story on the left, and the **bolded** sentence, a model should infer assertions such as the ones on the right.

Recall that we define an *assertion* as a tuple that represents a fact. This tuple contains at least a subject, a relation type, and an object (similar to subject-verb-object triples). We add a field to this tuple, which is *specificity*. We define *specificity* as whether the assertion's content is about specific entities in the text context, or if it is a generalized version of an assertion. This can be seen as whether the assertion is a *general* template with variables, or a *specific* instance of this template. In the case of a story, contextual commonsense inference can help with story understanding (e.g., a contextual commonsense inference system could infer assertions in a story that indicate a revenge plot [180]). In the more interesting and relevant use case of procedural understanding, it helps with step explanations and step rephrasing by giving possibly unstated assertions (e.g., a pen is required for the first step of writing a check which is writing the date).

Such framing additionally allows us to utilize a trained contextual commonsense inference model downstream by going sentence-by-sentence, inferring assertions as the context changes. To clarify the task of contextual commonsense inference even further, below we give an example with a story, **a target sentence**, and some corresponding story-specific and general commonsense assertion inferences. The story comes directly from the ROCStories corpus [111]:

> **Story:** The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! **They scored a final goal!**
>
> **Story Specific Commonsense Inference:** The red team, is capable of, winning the game
>
> **General Commonsense Inference:** Some people *scored a final goal* , causes, some people to be happy

This task is hard for modern pre-trained contextual language models [50]. This may be because it may rely on information that a model may not have seen during pre-training, or the model has to figure out what topic to infer information about. The first issue is exacerbated because commonsense knowledge, present in everyone and the target for a model trained in this task, tends to not be written explicitly in text [191, 92, 33, 36]. In addition to these problems, the correctness of the information that is generated by the models is hard to evaluate and usually involves a costly human-in-the-loop setup.

Prior work, such as COMET [17], has tried to do sentence-level commonsense inference: generating a commonsense assertion, with at most a sentence as context. ParaCOMET [50] is an extension of COMET that was developed to work at a paragraph level (i.e., what we describe as the contextual commonsense inference task). ParaCOMET utilizes a recurrent memory and is trained on a corpus of aligned stories and assertions. ParaCOMET builds this dataset to address the contextual commonsense inference task by aligning facts from a commonsense knowledge graph (i.e., ATOMIC [155]) with a story (i.e., sampled from ROCStories [112]) through a heuristic based on the ROUGE [90] metric. It goes a step further by utilizing the cross entropy of the tokens (story and a fact) pre-trained language model, as a measure of coherence to keep only assertion matches that are coherent to the narrative. They additionally address the need for memory (i.e., for the model to remember prior events) by using and saving prior aligned assertions in a memory system. An example of an input and expected output from ParaCOMET can be seen below:

> **Model Input:** The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! **They scored a final goal!** <|sent5|> <|xEffect|>
>
> **Model Target/Output:** win the game

We note that our broader work consists of applying this for procedural understanding. However, as we mentioned before, all procedures are stories, but not all stories are procedures. Early in this Chapter, we utilize the hockey game story as a guiding example of how we can apply contextual commonsense inference. However, at the end of the chapter, we switch to applying this contextual commonsense inference for procedural understanding.

In this example, since the model is predicting ATOMIC objects, the output is a single phrase (i.e., win the game). Additionally, the symbols *</sent5/>* and *</xEffect/>* mean that the target sentence is sentence number five[1], and that the relation we want to generate a tuple about is the "has the effect on a certain person(s)" respectively[2].

Another parallel work that has tackled contextual commonsense inference is GLUCOSE [114]. GLUCOSE annotates the ROCStories [112] corpus along ten dimensions of commonsense. The authors annotate every sentence with an assertion that is either

---

[1] We note that in the original ParaCOMET work, the sentences were 0-start indexed. We utilize 1-start indexing for clearer understanding.

[2] This textual explanation of the relation is taken from the original ATOMIC[155] work.

present or implied in it for a given dimension. Additionally, they annotate each assertion with a general version of it, as we defined previously as general inferences, which includes variables and their descriptions. What this means is that any person(s) or object(s) in the assertion is(are) replaced with a token such as *Person_A*, etc. to represent a "general" or templated version of the fact. An example of the GLUCOSE formulation's inputs and expected outputs is given below:

> **Model Input:** 1: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! ***They scored a final goal!****
>
> **Model Target/Output:** The red team scores, Causes/Enables, they win the game ** People_A score, Causes/Enables, People_A win a game

This formulation of contextual commonsense inference is harder than the Para-COMET one because it has to generate two sets of assertions a subject, relation, and an object tuples, where one is the story specific one and the other is the general version of the assertion. These are seen above, separated by the ** respectively. In this example additionally, we can see the symbol *1:* which tells the model to predict along a dimension of commonsense described by GLUCOSE (i.e., 1: Event that directly causes or enables X), and the sentence enclosed by asterisks (*) which signifies it is the target sentence. With this corpus of annotated stories, the authors train a T5 [142] model to, given a dimension, target sentence, and story, generate both a story-specific and general assertion. It is worth noting that in both works, the models have no way to control their inference; they have to do it using only the story, a target sentence, and a relation.

None of these works address controllability in the generation, which means that the models can generate assertions that may be irrelevant to the sentence, or may not be about a topic needed for a downstream application. Additionally, these models are only trained on one dataset at a time, which can hinder a model's capability to infer knowledge if it has not seen the knowledge elsewhere. Lastly, these models do not score the factuality or correctness of the assertions; at most they can generate a beam/sequence score, which indicates the likelihood of the generated phrase.

In this chapter, we attempt to address all of these shortcomings through various techniques. Firstly, we construct a dataset of contextualized assertions consisting of assertions from ConceptNet [163], ATOMIC 2020 [65], and GLUCOSE [114][3]. To construct this dataset, we align the ROCStories [112] with an assertion by generating sentence/paragraph embeddings for the stories and the assertions by using the sentence-transformers [146] library. We then use cosine distance to find the closest story for each assertion. With this closest story, we repeat the process once more, now with the sentences from the story, to find the closest sentence in the story to the assertion. This contextualization and alignment technique, while simple, puts all the knowledge bases in the same contextual universe. When we train models utilizing this aligned dataset, we call them models that perform *joint inference* (as we have joined multiple knowledge bases and are leveraging them for commonsense fact generation). We refer to this later on as *joint inference.* Secondly, we augment this dataset of aligned assertions, stories, and target sentences, with "hints", as a method to communicate constraints when performing contextual commonsense inference. We automatically generate "hints" by selecting parts of a target assertion, along with a symbol identifying the parts, and include them as input to our contextual commonsense inference model. Lastly, we use this dataset to train two language models; one to infer assertions from the story and target sentence, and a second to validate or score the assertion given the story and the target sentence. We also explore training these models in an adversarial manner. We utilize this set of models, in the downstream application of contextual procedural knowledge graph construction.

Altogether, our contributions in this area are:

- The utilization of a hinting mechanism to help condition and control a generative model for contextual commonsense inference.
- A simple method for contextualizing assertions to a given text with the purpose

---

[3] For our final model, which we utilize for procedural understanding, we additionally include the Ascent++ dataset [119]

of performing joint inference.

- A method for adversarial training of language models to infer and evaluate assertions from a story context.

4.3 Hinting for Controllable Generation

In this section, we detail a technique that we propose called "Hinting" to address the lack of controllability in contextual commonsense inference. Later on, we utilize this hinting technique in procedures to convert them from text into contextual procedure knowledge graphs.

4.3.1 What exactly is hinting?

Recently, there has been work on exploring *prompting* strategies [93] for pre-trained, transformer-based language models [171, 41]. These are methods that alter the input to a language model such that it matches or approximates templates that it has seen during pre-training and can reuse or exploit this information. Prompting helps achieve higher performance in tasks with less training data, can help with controllability in the case of text generation, and is more parameter-efficient and data-efficient than fine-tuning, in some cases [86]. One type of prompting is *prefix prompting* [86, 78]. Prefix prompting consists of altering a language model's input (i.e. prefix) by adding additional words. These words can be explicit hard prompts such as actual phrases or words, or they can be soft prompts, embeddings that are input into a model and can be trained to converge on some virtual template or virtual prompt that can help the model.

Prompting holds the potential for improving contextualized commonsense inference. We utilize the idea of a *hint*, a hybrid of hard and soft prompts. We define a *hint* as the part(s) of an assertion that a model has to predict, along with special identifiers for these parts, wrapped within parenthesis characters.

Syntactically, a *hint* takes the form of: "*([specificity], [subject symbol,subject], [relation symbol,relation], [object symbol,object])*" where the actual content of the hint between the parenthesis would be a permutation of all but one of the elements in the target tuple. Hints are provided during training by sampling a binomial distribution ($p = 0.5$) for each element in a minibatch, which determines whether to give a hint or not. The actual content of the hint would then be generated by random sampling without replacement up to all but one of the elements in a target tuple. In Table 4.1 we can see various examples of how the hinting mechanism works for a model trained for contextual commonsense inference on GLUCOSE.

Model Input	Model Output
7: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal!. * They scored a final goal!. *	They scored a final goal >Causes>They feel(s) happy ** Some People_A scored a final goal >Causes>Some People_A feel(s) happy
7: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal!. * They scored a final goal!. * **hint:** (<|**specific**|><|**subj**|>**the red team scores the final goal**)	**the red team scores the final goal** >Causes>the red team feel(s) happy ** Some People_A (who are a team) score the final goal >Causes>Some People_A feel(s) happy
7: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal!. * They scored a final goal!. * **hint:** (<|**specific**|><|**subj**|>**the blue team does not score the final goal**)	**the blue team does not score the final goal** >Causes>the blue team feel(s) disappointment ** Some People_A do not score the final goal >Causes>Some People_A feel(s) disappointment
7: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal!. * They scored a final goal!. * **hint:** (<|**specific**|><|**obj**|>a child)	they scored a final goal >Causes>**a child** feel(s) happy ** Some People_A scored a final goal >Causes>Someone_A feel(s) happy
7: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal!. * They scored a final goal!. * **hint:** (<|**general**|><|**subj**|> **Something_A (that is a point)**)	They scored a final goal >Causes>They feel(s) happy ** **Something_A (that is a point))** is scored >Causes>Some People_A feel(s) happy

Table 4.1: Example of inputs and outputs for the GLUCOSE trained contextual commonsense inference model with hints. The *hint* is **bolded** and the parts of the *hint* are colored (subject, relation, object). Without a hint we can see that the model tries to infer directly on the content of the sentence, however with *hints*, the model tries to include an inference based on the target sentence with the contents of the *hint*.

Here we can see more clearly that whenever we give a hint a model trained with hints (i.e., hinting) tends to produce generations that include the components given in the hint. We utilize hinting in training our models from here on out unless otherwise stated. The controllability that hinting enables can permit us to use models trained with it in downstream applications such as contextual knowledge graph generation.

An example of Hinting

A simple example of *hinting* is the following:

> **Story:** *The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! They scored a final goal!*
>
> **Target sentence:** *They scored a final goal!*

> **Target assertion:** *(subject: the red team, relation: are capable of, object: winning the game.)*

A hint can be any permutation of the target assertion, except the complete assertion, along with some symbol that indicates which part it is:

> **Possible Hints:** *(<|subj|> the red team), (<|subj|> the red team, <|rel|> capable of), (<|subj|> the red team, <|obj|> winning the game), (<|rel|> capable of, <|obj|> winning the game), (<|obj|> winning the game), (<|rel|> capable of)*

A hint for the given story, target sentence, and target assertion, yields the following:

> **Hint:** *(<|subj|> the red team, <|rel|> capable of)*

Putting everything all together, the input for the model would be:

> **Story with Hint:** *The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! They scored a final goal!* ***(<|subj|> the red team, <|rel|> capable of)****.*

We note that this is a generic version of how the hinting mechanism works, and individual datasets (i.e., ParaCOMET and GLUCOSE) have slightly different variations of this.

4.3.2 Discourse-aware/contextual commonsense inference

Recall that commonsense inference is the task of generating a commonsense assertion. Discourse-aware/contextual commonsense inference is the task of, given a certain context, inferring commonsense assertions that are coherent within the narrative [50]. This task is particularly hard because commonsense knowledge may not be explicitly stated in text [92] and the model needs to keep track of entities and their states either explicitly or implicitly. Research into the knowledge that pre-trained language models learn has yielded good results in that they do contain various types of factual knowledge, as well as some commonsense knowledge [33, 130, 36]. The amount of commonsense knowledge in these models can be improved by supplementing sparsely covered subject areas with structured knowledge sources such as ConceptNet [163, 36]. Knowing that these pre-trained language models may contain some commonsense information has led to the development of knowledge models such as COMET [17]. This line of research has been extended from the sentence-by-sentence level in COMET to the paragraph-level in ParaCOMET [50]. Contemporaneously, GLUCOSE [114] builds a dataset of commonsense assertions that are contextualized to a set of stories, and generalized (e.g., *John is a human* is generalized to *Someone_A is a human*).

The general task of contextual commonsense inference can be formally described as follows. We are given a story S composed of n sentences, $S = \{S_1, S_2, \ldots, S_n\}$, a target sentence from that story, S_t, where $S_t \in S$, and a relation type R. Given all this, we want to generate a tuple in the form of $(specificity, subject, R, object)$ that represents an assertion, present or implied, in S_t given the context S, and the relation type R.

4.3.3 Testing Hinting: Experimental Setup

We run two sets of experiments to show the effectiveness of hinting. The first is utilizing the original ParaCOMET dataset and setup and adding hints. The ParaCOMET setup consists of given a story S composed of n sentences, $S = \{S_1, S_2, \ldots, S_n\}$, a relation type R, and a target sentence token (i.e. $<|sent0|>$, $<|sent1|>$, ..., $<|sent(n-1)|>$). In the ParaCOMET dataset, we must predict the object of a triple, utilizing implicitly the sentence as a subject and explicitly the supplied sentence symbol and relation R symbol.

Within this framework, after the relation R, we add our hint between parenthesis (i.e. "($[hint]$)"). In this framing, our hint can be composed of: a subject symbol (<|subj|>) along with the target sentence to serve as a subject, a relation symbol along with the relation R, or an object symbol along with the object of the triple. Using the hockey example, a possible hint in this set of experiments would be: "(<|rel|> <|xEffect|>,<|obj|> they win the game)".

In our experiments on the ParaCOMET formulation with the GPT-2 model, we utilize the same cross-entropy loss as in [50]. We note that we utilize a sequence-to-sequence [166] formulation for the T5 and the BART models. This in contrast to the GPT-2-based system requires encoding a source sequence (i.e., story, target sentence, and relation symbol), and decoding it into a target sequence (i.e., the object of an assertion). For the T5 model, we add the prefix "source:" before the story S, and the prefix "hint:" for placing our hints. For simplicity, we construct the same "heuristic" dataset as ParaCOMET which utilizes a heuristic matching technique to align ATOMIC [155] triples to story sentences.

For our second set of experiments, we utilize the formulation utilized in GLUCOSE [114]. The formulation utilizes the T5 model in a sequence-to-sequence formulation once more. In this formulation, the source text is composed of a prefix of a dimension to predict $D \in 1, 2, \ldots 10^4$, followed by the story S with the marked target sentence. The target sentence, S_t, is marked with $*$ before and after the sentence. An example input is: "1: The first sentence. ***The target sentence.*** The third sentence.". This task is slightly different from the ParaCOMET one, in that in addition to predicting a context-specific assertion, the model has to predict a generalized assertion (i.e., in this task we have to infer a general and context-specific subject, object and a relation). For our hints, we provide up to five out of these six things, along with a symbol that represents whether it is the subject, object or a relation, and another symbol that represents whether it is part of the general or specific assertion. We add our *hint* after the story S, utilizing the prefix "hint:" and supplying the hints between parenthesis.

We run the ParaCOMET experiments for 10 epochs on the dataset's training data and evaluation data. We utilize a max source sequence length for the BART and T5 models of 256, and a max target length of 128. For the GPT-2 models, we utilize a max sequence length of 384. Additionally, we use the ADAM [75] optimizer with a learning rate of 2e-5, and a linear warm-up of 0.2 percent of the total iterations. For the T5 models, we utilize a learning rate of 1e-4 because early experiments showed

---

[4] The definition for each dimension number is given in the GLUCOSE work. Dimensions in GLUCOSE are (explicit or implicit) relations that help explain causality between the entities mentioned.

that the model would not converge with lesser learning rates. We utilize the scripts from [50] for data generation. We also utilize a batch size of 4 for training and we accumulate gradients for 4 steps for an effective batch size of 16. The results that we present are the average of the 10 runs over 4 seeds for hinted and non-hinted conditions.

We run GLUCOSE experiments for 5 epochs and 4 seeds on the original GLUCOSE data. Additionally, we utilize a linear warm-up of 3000 steps. We utilize the ADAM optimizer with a learning rate of 3e-4, a train batch size of 4, with gradient accumulation of 4 steps for an effective batch size of 16, and a max source length of 256 and a max target length of 128. In our results, we present the average of the 4 seeds across the 5 epochs. In both experiments we report the scores given by SacreBLEU [134], ROUGE [90], and METEOR [10] using the datasets library [81] metrics system. We run our experiments in a machine with an AMD ThreadRipper 3970 Pro and 4 NVIDIA A6000s. Every epoch per model is approximately an hour.

Additionally, we run a small Mechanical Turk study similar to the one presented in the original ParaCOMET [50] in which a human judges a generated assertion and judges the plausibility of it on a 5-point Likert scale: obviously true (5), generally true (4), plausible (3), neutral or unclear (2), and doesn't make sense (1). We present the results in the same manner where Table 4.4 displays the percent of inferences judged as plausible or true (3-5) and the average rating per inference. Participants were given \$0.1 to complete the task. We sample from each of the ParaCOMET and GLUCOSE test sets, 100 entries. Then based on the models for each dataset, we pick the epoch that had the highest automated scores and we proceed to randomly sample one of the trained hint and non-hinted models. We then select one sentence from the randomly sampled test entries and ask both models to generate an inference along a randomly sampled relation or dimension for that sentence.

Model	BLEU		METEOR		ROUGE1		ROUGE2		ROUGE L		ROUGE L SUM	
	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint
ParaCOMET	**42.705***	41.960	**59.411***	59.045	**63.339***	61.454	**52.483***	50.513	**63.292***	61.395	**63.294***	61.399
Bart	**41.765***	41.639	**58.766***	58.639	**61.054**	61.013	**49.970**	49.889	**61.004**	60.964	**61.010**	60.969
T5	41.070	**41.102**	**58.004**	58.000	59.535	**59.631**	48.695	**48.823**	59.488	**59.588**	59.494	**59.597**

Table 4.2: Averages of 4 different seeds over 10 epochs for *hinted* (Hint) and non-hinted (No Hint) runs of the ParaCOMET dataset from [50]. The largest scores are **bolded** and significantly different scores have an asterisk (*) next to them. We can see from the results that *hinted* systems tend to achieve higher performance even if slightly and in some cases significantly, and do not decrease performance significantly. For significance we use the t-Test: Paired Two Sample for Means.

BLEU		Meteor		Rouge 1		Rouge 2		ROUGE L		Rouge LSUM	
No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint
58.542	**59.099***	66.829	**66.917**	66.387	**66.681***	47.850	**48.141**	62.542	**62.874***	62.528	**62.868***

Table 4.3: Averages of 4 different seeds over 5 epochs for *hinted* (Hint) and non-hinted (No Hint) runs of the GLUCOSE contextual inference task dataset. This is the same dataset as the work in [114] The largest scores are **bolded** and significantly different scores have an asterisk (*) next to them. Once more, we see that hinting provides a small, increase in performance, all the while permitting controllability. We use the t-Test: Paired Two Sample for Means for significance

4.3.4 Results and Effects of hinting

Experiment 1: ParaCOMET with *hints*

The aggregated (averaged) results for this set of experiments can be found in Table 4.2. We can see here that on average, hinting does tend to improve the score even if slightly. It seems that providing a hint is beneficial and not detrimental for contextual commonsense inference. Given the way that this task is framed, a possibility that could explain the relative similarity of the performances is that hinting *only* adds the object of the assertion as additional possible data that the model may see during training; the subject and the relation can be repeated with hinting. We note that the performance of the T5 model was less than that of the other models, and we believe that it may be a lack of hyperparameter tuning, as it was seen that the model was sensitive to the learning rate and had to use a higher than usual learning rate.

Experiment 2: GLUCOSE with *hints*

The aggregated results for this set of experiments can be found in Table 4.3. Once more, we notice that hinting does tend to improve the performance of the contextual

Model	Non-Hinted	Hinted
ParaCOMET	3.71	**3.76**
Bart	**3.72**	3.48
T5	**3.73**	3.68
T5-GLUCOSE	**4.10**	4.06

Model	Non-Hinted	Hinted
ParaCOMET	81%	**84%**
Bart	**83%**	74%
T5	**81%**	**81%**
T5-GLUCOSE	**92%**	90%

Table 4.4: Results of human evaluation of ParaCOMET and GLUCOSE datasets. The largest scores are **bolded** and significantly different scores have an asterisk (*) next to them. We sampled 100 test points for each model from their test datasets and had the hinted and non-hinted models infer assertions. Humans judged these assertions on a 5 point Likert scale where above 3 was plausible similar to [50]. On the left we can see the average values of the human judgments and on the right we can see the percentage of plausible inferences (rated >= 3). We can see that hinting provides comparable performance.

commonsense inference task. This suggests that hinting is indeed beneficial for the task, especially when faced with the harder task of generating both a general and specific assertion. We believe that this improvement is because hinting gives the model the clues it may need to decide on what to focus or attend to, to generate useful inferences, but further experimentation would be needed to verify this.

Experiment 3: Human Judgements

The results of the small Mechanical Turk study for human evaluation of model inferences can be seen in Table 4.4. Overall, we can see here that hinted systems are judged as less plausible. Interestingly, after inspecting the results where there was a large difference (more than two points between the systems), we see that there are some cases in which the same or very similar responses got completely different scores. We also see upon looking at some of the inferences that the hinted model tends to be more general and provide shorter responses than the non-hinted model (e.g., hinted inference: “satisfied” vs. non-hinted inference: “happy and satisfied”).

4.3.5 Final Remarks on Hinting

From the results of our experiments, we can see that hinting tends to increase the performance of contextualized commonsense inference at least with regard to automated metrics, and does not significantly degrade or improve human judgments. Without any significant cost, by utilizing hinting, we gain controllability in the generation. By supplying these*hints*, we are teaching the model to pay attention and generate inferences about a certain subject, relation, or object. This in turn, after training, can be leveraged by a user or downstream application to guide the model to generate assertions from parts that are manually supplied. Although this is not very clear within the ParaCOMET formulation, it becomes clearer in the GLUCOSE formulation of the problem. We give an illustrative example of the usefulness of hinting in Table 4.1. We can see that by giving a model the hint, the model could be capable of inferring information that may not be present in the story. We note that this behavior is useful in downstream tasks such as story understanding and contextual knowledge graph generation, in which we may need a model to have a specific subject or object.

4.4 Joint Inference of Commonsense Assertions

Given that we have presented hinting, which is a method to help with the controllability of contextual commonsense inference generation, we now look at how we can combine multiple knowledge bases for this task along with hinting.

4.4.1 What is joint commonsense inference?

In this work, we define *joint commonsense inference* as inferring commonsense knowledge assertions by leveraging knowledge from multiple knowledge bases. To illustrate this, we give the following example, story:

> *John is a regular person who has a dog.* ***John, every day, goes out to walk his dog.*** *John met a friend when walking his dog. They exchanged stories about their dogs.*

From this story, we want to infer the general version of the commonsense assertion of "John is capable of walking his dog", derived from **the second sentence**. This general version can look similar to "Someone_A who has an animal (that is a dog) enables Someone_A to walk the animal (that is a dog)". To generalize this, we must know that: *John is a person's name*, which we can find from a semantic tagger. A much more commonsensical fact needed to infer the assertion is that: *a dog is an animal*, which is a fact found in ConceptNet. Lastly, to infer the assertion, we need to know that: *A person having a dog has the effect that a person goes to walk their dog*, which is a fact that we could find from ATOMIC 2020. Therefore, to infer the general assertion that we presented, we must join information from *at least* two knowledge bases to infer our general assertion. This process of joining the information from multiple sources is what we call *joint commonsense inference.* Joint commonsense inference is useful because it could lead to implicitly applying or combining knowledge and/or analogies that might be present in the different knowledge sources, which may lead to better results when performing contextual commonsense inference.

4.4.2 Joint Inference Approach Overview

To perform joint inference in the task of contextual commonsense inference, we propose the following approach:

1. *For each knowledge base that we have, we convert each of the assertion found in them into a tuple format of* ***{subject, relation, object, specificity}***[5]*. We note that each part of the tuple must be text*[6] *(i.e., if the symbolic version of a relation is "IsA", the textual version would be "is a").*
2. *We align each knowledge base tuple with a story (e.g., the ROCStories corpus) and target sentence from the story. The target sentence is the sentence that is most likely to be used to infer the tuple. We perform the alignment by vectorizing the tuples and stories and utilizing nearest neighbors with the cosine distance as a metric. We give details of this alignment in section 4.4.4.*

---

[5] This follows a similar pattern to subject-verb-object triples, but has the added field of specificity which is whether the assertion is contextual to a story, or a generally applicable assertion

[6] This ultimately helps us express the assertion in a textual way (i.e., (a dog, IsA, animal) when converted to the tuple (a dog, is a, animal, specific) and passed to a string representation function can be expressed as "Specifically, a dog is an animal".

3. *We combine all of the aligned assertions and context into one list and shuffle it.*
4. *We replace the naming scheme of variables that may be present in general assertions with the naming scheme from GLUCOSE (e.g., PersonX is replaced with Person_A).*
5. *We train a contextual commonsense inference model on this dataset, whose inferences are joint inferences.*

By following this procedure, we will end up with a dataset of story-aligned assertions. In this dataset, all of the assertions are grounded in the same set of stories. With this, we can train models that can perform joint contextual commonsense inference. Now we will go into some details of this process.

4.4.3 Specificity in assertions

Recall that we define *specificity* as whether an assertion's content is about specific entities in the aligned story, or if it is a generalized version of an assertion. This can be seen as whether the assertion is a *general* template with variables, or a *specific* instance of this template. To make the difference between *specific* and *general* assertions clearer, we give the following example. Using the same story as before:

> *John is a regular person who has a dog.* ***John, every day, goes out to walk his dog.*** *John met a friend when walking his dog. They exchanged stories about their dogs.*

As before, we focus on the second sentence: **John, every day, goes out to walk his dog**. From here, we can infer the *specific* assertion: "John is capable of walking his dog". The assertion is *specific* because it fills out a broadly applicable template, which we will present next, based on the story context about John and his dog. From the sentence, we can also infer the *general* or templated version of the assertion: "Someone_A who has Something_A (that is a dog) enables Someone_A to walk the Something_A (that is a dog)". This latter assertion is *general* because it states the assertion in a template format (i.e., broader terms) the same fact. A *general* assertion is not the story-dependent instance of the template, but the broader, story-independent template. These *general* assertions contain variables in them.

In this section, we describe our approach for joint inference in which we utilize ConceptNet, ATOMIC 2020, and GLUCOSE as our knowledge bases. We start by looking at the specificities in these KBs. In Table 4.5 we give the different available specificities for these knowledge bases. From this, we can see that ConceptNet *does not* have *general* specificity assertions. Although this may sound counterintuitive, ConceptNet gives *specific*, untemplated, instances of assertions, in contrast to ATOMIC and GLUCOSE, which describe *general* versions of assertions. ATOMIC 2020 has the opposite problem, it gives *general* versions of rules, (e.g., PersonX participates in some event, has some effect on PersonX or Y around them), and does not give, within our contextual commonsense inference framework, the *specific* instance of the templates (e.g., filling out PersonX, PersonY, etc.). To remedy this lack of specificity within two of our sources, we mention ways to generate examples of the missing specificity and implement the solution for ATOMIC 2020.

Knowledge Base	**General**	**Specific**
ConceptNet	✗*	✓
ATOMIC 2020	✓	✗*
GLUCOSE	✓	✓

Table 4.5: Here we can see the available specificities in ConceptNet, ATOMIC 2020, and GLUCOSE. We mark with ✗ the specificities that are not available by default, and add * to those that can be generated.

Generating Missing Specificity

ConceptNet To generate *general* assertions for ConceptNet, one possibility is to run a classifier that would determine whether a given set of tokens is a person, place, or object, among others. With this information, we could fill out, as an example, the template that GLUCOSE broadly utilizes which is: {Category}({Description}), relation, {Possibly Other Category} ({Possibly Other Description}). From ConceptNet, we could find the relation: “a dog, IsA, animal”. A *general* version of this assertion

Replacement Type	**General Assertion**	**Generated Specific Assertion**
Instantiate Person Variable	**PersonX** has a dog has the effect on **PersonX** they are happy	**John** has a dog has the effect on **John** they are happy
Fill in the blanks	PersonX absolutely loved ____ makes PersonX have the attribute compassionate	PersonX absolutely loved **his dog** makes PersonX have the attribute compassionate

Table 4.6: Here we see two examples of instantiating general information based on the context of the story of John and his dog to generate story-specific information.

can be "Something_A (that is a dog), IsA, Something_B (that is a animal)". Although we describe this process, we do not implement it in our work and leave it as future work.

ATOMIC 2020 To generate *specific* assertions for ATOMIC 2020 we can do the following. We can first identify variables (PersonX, PersonY, etc.) that are in the ATOMIC-2020 assertion. We can then replace these variables with a Mask token from a language model that was trained with the masked language modeling objective [41, 94], and use the language model to fill in the Mask token similar to a Cloze [169] (i.e., fill-in-the-blanks) task. To give the model sufficient context, we insert the assertion to the right of the nearest aligned sentence (we describe the process to get this in the next section). In the case of Person$N$ (e.g., PersonX, PersonY, etc.) variable, this usually leads to the variable being replaced with a character from the story. In addition to this, ATOMIC 2020 contains blanks demarcated by underscore characters (i.e., ____), which we can once more replace with a Mask token and have the model fill it out with the given context. We use this process in our work, filling in the blanks with a ROBERTA [94] large model. An example of generating the specificities can be seen in Table 4.6.

4.4.4 Aligning Assertions with Stories

To align assertions with stories, we do the following procedure. On a high level, we vectorize the stories, we vectorize the assertions and we then utilize the cosine distance to find the nearest story for each assertion. We then go into more detail and repeat the same procedure (i.e., vectorization and similarity search) for each sentence in the previously found nearest story. Ultimately, we are left with the story and sentence that is most relevant or similar to the assertion. On a low level, we utilize the sentence transformers package along with the “paraphrase-mpnet-base-v2” model from the repository, to generate a representative vector for every story and for every assertion from each of our knowledge bases. We then utilize the FAISS package [70] to perform a fast approximate cosine similarity search to find, for each assertion, what is the nearest story. Once we have this nearest story, we again utilize the sentence transformers model to vectorize every sentence in that story along with the FAISS package for the cosine similarity search, to find the nearest sentence to the assertion. This process can be visualized and figures 4-2 and 4-3.

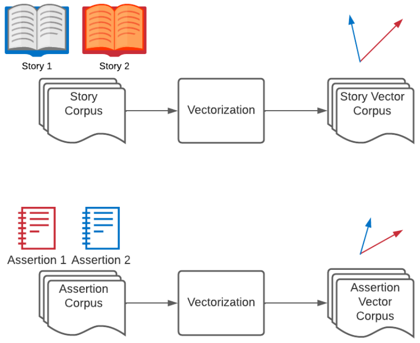


Figure 4-2: Step 1: The story and assertion corpus are vectorized. In our work we utilize the sentence-transformers package [146] to achieve this.

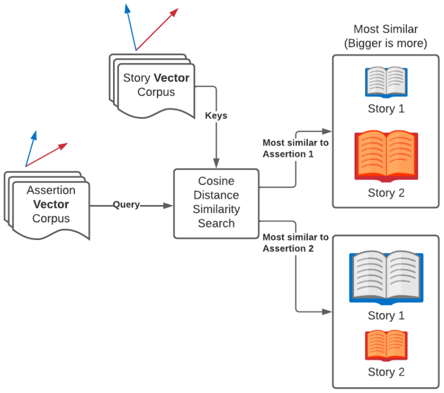

Story **Vector** Corpus
Keys
Most Similar
(Bigger is more)
Story 1
Story 2
Most similar to Assertion 1
Assertion **Vector** Corpus
Query
Cosine Distance Similarity Search
Most similar to Assertion 2
Story 1
Story 2

Figure 4-3: Step 2: The resulting assertion vectors are utilized as queries, and the resulting story vectors are used as keys for a memory-like lookup. In this work, we use the FAISS package for this. The output of the memory-like lookup is the nearest story for each vector. This process is repeated for the sentences in the nearest story, to align the assertion with a sentence.

4.4.5 Experimental Setup

To evaluate the effects of joint inference by combining multiple knowledge bases in the task of contextual commonsense inference we do the following. We generate a story-aligned assertion dataset for each knowledge base individually (i.e., for Concept-

Net, for ATOMIC 2020, and for GLUCOSE) as described in the previous sections. Once we have generated a dataset for each, we proceed to perform combinations of the datasets: ConceptNet-ATOMIC 2020, ConceptNet-GLUCOSE, GLUCOSE-ATOMIC, and ConceptNet-ATOMIC-GLUCOSE. For the individual and the combined datasets, we perform three sets of automated tests. One that includes hinting the specificity, subject, and relation during evaluation, one that includes hinting the subject during evaluation, and the other without these.

Although this may seem counterintuitive and like "cheating", the hinting gives us a mechanism to be able to force the model to predict assertions in the format of the knowledge base that they were pulled from (i.e., we would use certain relations such as "located at" in ConceptNet for ConceptNet inferences, and "results in" for GLUCOSE inferences). Expanding on this, the rationale behind these setups is that we want to evaluate what the model infers without any guidance, and see what it infers with varying levels of guidance with multiple knowledge sources. To train our joint inference models, we use a batch size of 50, on 4x NVIDIA A6000, a learning rate of 1e-5 for an ADAM [75] optimizer, and 3 epochs over all the contextually aligned assertions.

We note that the data for ConceptNet we utilize is the dataset given by [85], specifically the data in the "train_600k.txt" which are approximately 600,000 examples of assertions from ConceptNet, and as a test set we utilize the "test.txt" that they provide. For ATOMIC 2020 we utilize the training and testing data files provided by the authors [65]. Lastly, for GLUCOSE we use the training and evaluation files also provided by the authors in the corresponding repository.

Additionally, we look into running a small Mechanical Turk evaluation of generated test assertions, because we suspect that automated metrics may hurt the model's evaluation when not using hinting. We sample 100 entries from the testing files of each knowledge base (ATOMIC 2020, ConceptNet, and GLUCOSE), and run these through a set of models trained firstly with only one of the test knowledge bases (i.e. a model trained only ConceptNet, a model trained in ATOMIC 2020, and a model trained in GLUCOSE) and secondly a model trained with the combination of knowledge bases and evaluated with and without hinting. We take the generated inferences and ask 2 raters from Amazon Mechanical Turk to determine whether the assertion is acceptable, whether it is acceptable with the context that it was aligned with, and whether the gold standard assertion was acceptably aligned with the context. We mark as acceptable the answers that both human annotators agree as valid and anything else as invalid.

4.4.6 Effects of joint inference

The results for our automated experiments can be found in Table 4.7 and from our human experiments in 4.8. From our experiments in this area, we notice the following. Firstly, when training with hinting, joint inference *does not* seem to improve the performance of synthetic tests. In other words, without any hints, and with multiple knowledge sources, we do not get any large differences in automated metrics. What this may mean is that when we utilize hinting effectively, it may manage to utilize the format (e.g., relation types) of the knowledge base that it is tapping into for information. Additionally, some of the knowledge sources that we are using have a little overlap (GLUCOSE and ConceptNet had approximately 0.34% of overlap [114], and ATOMIC 2020 has approximately 9.4% of overlap with ConceptNet [65]), which means that once hinting is utilized to give control signals to the models, this lack of overlap may attribute to why the metrics do not decrease drastically. Secondly, without hinting, in the automated tests that we run, performance seems to degrade when we add knowledge bases. Upon further inspection of the results, the reason for this seems to be that the model thinks that an assertion in the format of another knowledge base (e.g., generalized assertions from GLUCOSE on the ConceptNet test set, relation type from ATOMIC in the GLUCOSE test set) may be more relevant than the test assertion types that we are evaluating. Upon inspecting some of the test results, this seems to be the case. We give one such example now, where a model that was trained on all the knowledge bases (ATOMIC 2020, ConceptNet, GLUCOSE) has to predict assertions for a test set from ConceptNet:

Training Set(s)	Test Set	Hint	BLEU	METEOR	ROUGE
ATOMIC 2020, ConceptNet, GLUCOSE	ATOMIC 2020	No	51.114	52.224	52.681
ATOMIC 2020, ConceptNet	ATOMIC 2020	No	51.164	52.151	52.713
ATOMIC 2020	ATOMIC 2020	No	51.139	52.699	52.904
ATOMIC 2020, ConceptNet, GLUCOSE	ATOMIC 2020	Subject	79.89	80.956	82.927
ATOMIC 2020, ConceptNet	ATOMIC 2020	Subject	80.046	81.144	83.087
ATOMIC 2020	ATOMIC 2020	Subject	80.203	81.221	83.079
ATOMIC 2020, ConceptNet, GLUCOSE	ATOMIC 2020	Subject, Specificity, Relation	87.031	88.172	89.606
ATOMIC 2020, ConceptNet	ATOMIC 2020	Subject, Specificity, Relation	87.091	88.242	89.645
ATOMIC 2020	ATOMIC 2020	Subject, Specificity, Relation	87.095	88.226	89.621
ATOMIC 2020, ConceptNet, GLUCOSE	ConceptNet	No	51.892	58.803	60.302
ATOMIC 2020, ConceptNet	ConceptNet	No	56.136	62.114	63.532
ConceptNet, GLUCOSE	ConceptNet	No	59.285	63.685	65.65
ConceptNet	ConceptNet	No	60.63	64.653	66.46
ATOMIC 2020, ConceptNet, GLUCOSE	ConceptNet	Subject	76.404	79.484	78.413
ATOMIC 2020, ConceptNet	ConceptNet	Subject	76.7	79.614	78.617
ConceptNet, GLUCOSE	ConceptNet	Subject	76.014	78.664	77.841
ConceptNet	ConceptNet	Subject	76.635	79.108	78.314
ATOMIC 2020, ConceptNet, GLUCOSE	ConceptNet	Subject, Specificity, Relation	92.695	94.253	94.171
ATOMIC 2020, ConceptNet	ConceptNet	Subject, Specificity, Relation	92.892	94.286	94.378
ConceptNet, GLUCOSE	ConceptNet	Subject, Specificity, Relation	92.729	94.071	94.109
ConceptNet	ConceptNet	Subject, Specificity, Relation	92.77	94.159	94.232
ATOMIC 2020, ConceptNet, GLUCOSE	GLUCOSE	No	36.23	41.338	48.629
ConceptNet, GLUCOSE	GLUCOSE	No	37.823	41.997	49.577
GLUCOSE	GLUCOSE	No	42.51	47.186	53.856
ATOMIC 2020, ConceptNet, GLUCOSE	GLUCOSE	Subject	80.879	82.014	85.664
ConceptNet, GLUCOSE	GLUCOSE	Subject	81.349	82.775	85.926
GLUCOSE	GLUCOSE	Subject	80.928	82.076	85.681
ATOMIC 2020, ConceptNet, GLUCOSE	GLUCOSE	Subject, Specificity, Relation	85.721	87.433	90.04
ConceptNet, GLUCOSE	GLUCOSE	Subject, Specificity, Relation	85.72	87.518	90.034
GLUCOSE	GLUCOSE	Subject, Specificity, Relation	85.65	87.473	89.967

Table 4.7: Here we present the results of our joint inference tests. We color code sets of rows as a testing run on ATOMIC 2020, GLUCOSE, and ConceptNet. The Training Set(s) column contains the knowledge bases that were used to train the model. The Test set column contains which knowledge base test set was used to evaluate the models. The Hint column represents the Hints that were given to the model during testing. Overall, we can see that with hinting on the test set (i.e., hinting the subject or the subject and the relation type), the addition of knowledge bases for inference does not improve nor degrade substantially the performance. To view this in the table, we can compare the rows in which the Hint column is either "Subject" or "Subject, Relation, Specificity". Additionally, we can see that without hinting on the test set (i.e., rows that Hint is "No"), the addition of knowledge bases for inference tends to decrease the performance.

> **Model Input: Someone I went to school with had a funny laugh.** He would chuckle. Then he would make a whew sound. He had not realized it before. He could not stop his crazy laugh.
>
> **Expected Output:** <general> <subject> person <relation> is/are capable of <object> laugh at joke

Model	**Acceptable**	**Contextually Acceptable**	**Alignment Acceptable**	**Acceptable**	**Contextually Acceptable**	**Alignment Acceptable**	**Acceptable**	**Contextually Acceptable**	**Alignment Acceptable**
ATOMIC 2020 - No Hint	0.70	0.66	0.6	-	-	-		-	-
ConceptNet - No Hint	-	-	-	0.77	**0.71**	**0.72**	-	-	-
GLUCOSE - No Hint	-	-	-	-	-	-	**0.81**	**0.68**	**0.8**
ATOMIC 2020 - ConceptNet-GLUCOSE - No Hint	**0.76**	**0.68**	**0.63**	**0.83**	0.7	0.65	0.79	0.67	0.59
ATOMIC 2020 - ConceptNet-GLUCOSE - Hint	0.71	0.53	0.57	0.77	0.64	0.68	0.77	0.64	0.68

Table 4.8: Results for human annotation of 100 randomly sampled assertions from ATOMIC 2020, GLUCOSE, and ConceptNet test sets and the inferred commonsense from these. We have three sets of three columns Acceptable, Contextually Acceptable, and Alignment Acceptable. Each set of columns is color-coded to represent a knowledge base. Firstly, the Acceptable column is the ratio of whether humans thought that inferred assertions, without context, were acceptably commonsense or not. The Contextually Acceptable, column represents the ratio of whether humans thought that inferred assertions given the context, were acceptable or not. Lastly, the Alignment Acceptable column is whether humans thought that the gold standard (from a knowledge base) assertion was correctly matched to the context. We can see that without hinting, the joint inference model (i.e ATOMIC 2020 - ConceptNet - GLUCOSE - No Hint) improves the acceptability, both with and without context, of assertions predicted in the ATOMIC 2020 test set. We can also see that performance does not degrade much in whether it produces assertions that are contextually acceptable throughout the test sets. We can see that with hinting, however, the performance is decreased and becomes closer to what the individually trained models can achieve. This suggests that with hinting, the model tries to channel the knowledge base that we are targetting, and aligns with what we see in the automated metrics.

> **Model Output:** <general> <subject> Someone I went to school with had a funny laugh. <relation> makes others react <object> entertained

> **Model Output with Subject, Relation, Specificity Hint (<general> <subject> person <relation> is/are capable of):** <general> <subject> person <relation> desires <object> laugh at joke

The relation “makes others react” is not from ConceptNet, but from ATOMIC 2020. If we hint the model to the subject, relation type, and specificity, it produces a result similar to what we are expecting, where it only defers to using the “desires” relation for the contextual inference. Which, in this case, would not be an incorrect inference.

Following this, we look at the results of our Mechanical Turk study, which can be found in Table 4.8. We can see that without hinting, the joint inference model (i.e. ATOMIC 2020 - ConceptNet - GLUCOSE - No Hint) improves the acceptability, both with and without context, of assertions predicted in the ATOMIC 2020 test set. We can also see that performance does not degrade much in whether it produces assertions that are contextually acceptable throughout the test sets. We can see that with hinting, however, the performance decreases and becomes closer to what the individually trained models can achieve. This suggests that with hinting, the model tries to channel the knowledge base that we are targeting, and this aligns with what we see in the automated metrics. This is reinforced by the test example given previously, and similar examples can be found for the different test sets.

Now, taking these results together, when we use hinting and join our multiple knowledge sources to perform this joint inference, we are able to within one model, essentially fit all the knowledge bases, that we are evaluating, at hardly any loss in plausibility/acceptability, or at the cost of automated metrics. This has implications for downstream applications because they no longer require multiple models. With one model and hinting we can do what three separate models would do. We also note that the performance that we get, is based on the "base" version of the BART model. Presumably, with larger models, the performance will increase adequately.

Lastly, we also note that on average our alignment technique has 60% approval rate for ATOMIC 2020, 68.3% for ConceptNet, and 69% for GLUCOSE, which gives us on average 65% approval for our alignment strategy of using sentence-transformers with the FAISS similarity search, which means that through our automated method, more than half of the assertions are presumably aligned correctly. We note that there are thresholds of similarity that can be fine-tuned to possibly improve this approval rating.

4.5 Adversarial Language Models

4.5.1 Joint and Adversarially training language models

Lastly, we provide and demonstrate the usefulness of a method for joint and adversarially training language models for the task of contextual commonsense inference. In the broader literature of generative adversarial networks (GANs) [54], the adversarial training of models, tends to lead to better results than training each model individually, possibly because of the gradients flowing from the discriminator informing the generator on how to improve. Additionally, the discriminative model can give a measure (usually in a 0-1 range) of how plausible the generations are. We propose using the generative language model (i.e., a generator) that we trained for contextual commonsense inference and combining it with a discriminative model (i.e., discriminator) whose inputs are the same story and target sentence along with the generator's inference. The discriminator can give a measure of how plausible the generated inference is for the given context. This general architecture can be seen in Figure 4-4.[7]

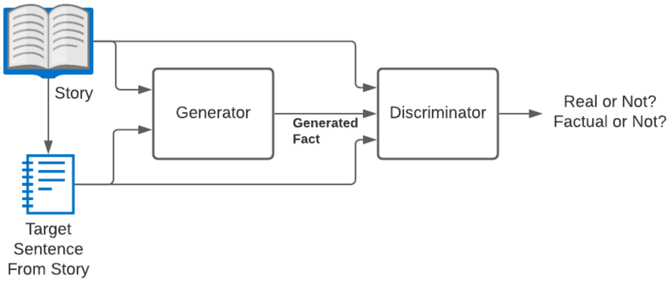


Figure 4-4: Overview of the proposed GAN architecture. A story and a target sentence are fed into the generator, which infers a contextual commonsense fact. This fact, along with the story and target sentence, is passed into a discriminator to determine whether it is from a generator or not and whether it is factual or not.

To be able to achieve this architecture, we need to be able to connect a generative language model to a regular language model with some additional final layers that produce a score. In our work, we utilize a transformer-based encoder-decoder generative model. Specifically, we use the BART model [79] for conditional generation, provided by the Huggingface Transformers library [182]. For our discriminator, we utilize a regular BART for sequence classification model also from Huggingface Transformers. However, it is not as simple as conditionally generating, and passing into the discriminator the generated assertions. The generative process utilized is re-

---

[7] One interesting aspect of this formulation, is that it becomes a kind of conditional GAN [101], which could reinforce the control signals given in hints.

current next-token generation. Recall that to pick the next token with this method, a non-differentiable *argmax* operation is used. This impedes the gradients from being calculated in backpropagation. The issue becomes even more complex, in that the generation process can utilize beam search to find even better generations, and each beam at the end of each generation step selects the next best token also with an *argmax*. To address this discontinuity, we utilize an approximation of the *argmax* (i.e., a soft *argmax*) described in the next section and similar to the work described in Section C.1.2, and perform a dot operation on the scores from this soft-*argmax* with the embeddings from the embedding layer to get an approximate and differentiable input embedding for the discriminator. Finally, we pick two of the same types of base model (e.g., BART), in order for both the generator and discriminator to share a vocabulary. The reason for sharing the vocabulary is addressed in the next section, however, this may not be necessary, and we give an alternative way of being able to "splice" together different models for this task in section 4.5.4.

We give some formal notations to describe the GAN framework. Let G be a learnable function (implemented as a Generative Neural model) that can take an input from a domain X and convert it to an output in another domain Y, namely $G : X \to Y$. That output Y is evaluated by a learnable function D that scores the output Y. A generative adversarial network (GAN) is an interplay between G and D, in which G tries to minimize the difference between what it generates, and D tries to maximize its discrimination of fake generations [54]. We note that we are not the first to attempt utilizing GAN systems for text generation as seen in Section C.1.2, but we are the first to apply this system to the task of contextual commonsense inference.

4.5.2 Addressing the Discontinuity in Generation

Recall that during recurrent conditional language generation, a next token, N, is selected by finding the *argmax* of a $softmax$ of all the vocabulary, after a language model is given the generated phrase up until step $N-1$. This is also called "greedy decoding". Also recall that an embedding layer is a neural network component that given an index i, returns a row vector, from a vocabulary matrix, that corresponds to i. This lookup operation can also be achieved by performing a dot product of a one-hot vector that represents the index i and the vocabulary matrix. This essentially scales every row in the matrix by the corresponding vector component and sums all the vectors. In the literature, this is also referred to as the Gumbel-Softmax [67]. In the case of a one-hot vector, it scales all but one vector to zero, therefore leaving only the desired i at the end of the summation.

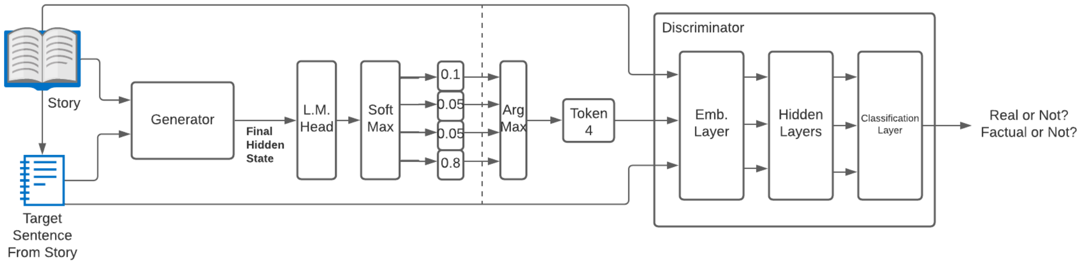


Figure 4-5: We visualize an example that shows the discontinuity when combining a generative language model with a discriminative language model. The dashed line represents where the gradients are discontinued because of the non-differentiable *argmax* operation.

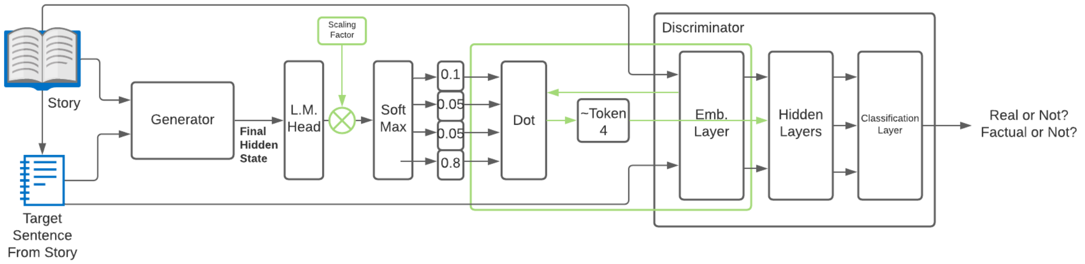


Figure 4-6: We visualize an example that shows how we address the discontinuity by replacing the non-differentiable *argmax* with a dot product between the softmax and the embedding layer matrix. Additionally, we highlight where the scaling factor is inserted to make the approximation more accurate. We mark our approach in green.

Now, to maintain the gradients, we need to connect the output of our generative model, which is the softmax, to the embedding layer of our discriminator model so that it can be input, scored, and backpropagated correctly. To do this, we simply replace the aforementioned one-hot vector that represents our index, with the softmax that the generative model produces at a given generation step, and perform a dot product of this softmax with the embedding matrix. This method is approximate, given that there may be noise from other non-zero elements in the softmax, and the top element is not an exact 1. To somewhat remedy this approximation, we can multiply the input of the softmax by a certain factor to essentially give a more approximate one-hot vector. However, this factor cannot be very large, because it may cause instability during backpropagation. In our work, we use a scaling factor of 1, as this seems to be accurate enough. We repeat this approximation for every generation step, and are left with a list of input embeddings for the discriminator that represent the output of the generator with usable gradients. Since we are using the same vocabulary, we can verify how accurate the output is, by training a K-Nearest Neighbors system, and finding the K=1 neighbor of the output of the softmax and embedding matrix dot product against the embedding matrix. We can use this test to determine an appropriate factor for scaling the softmax. Altogether, this approximation permits us to train our two language models adversarially by having gradients flow from the discriminator to the generator. We note that there is work that utilizes a similar simplification to permit gradient flow in an adversarial system in Section C.1.2.

4.5.3 Addressing Different Generation Types

The aforementioned approximation for the discontinuity, as we described it, can be utilized for greedy selection of the next-token (i.e., we always pick the maximal one from the final softmax). We can also apply this technique to beam-search generation, and at every point in constructing the top scored beam, we utilize the softmax of the maximal scoring beam, essentially simplifying the problem back down to a greedy generation-like formulation. In this work however, we do not explore top-k generation, top-p generation, nor sampling during generation. Top-k and top-p generation can be seen as masking out with zeros, tokens that do not meet a certain criteria. Sampling is more complicated. To use our approximation with sampling, we would need to model the sampling function at every generation step with something like a recurrent neural network. We leave this line of research for future work.

4.5.4 Splicing different models

We come back to address the issue of having to utilize models that have the same vocabulary. The reason for this is that the soft-argmax operation matches in matrix multiplication dimensions between models. We now give an alternative, although unexplored, option. Given that a generative model will produce a softmax vector of vocabulary size V, and we have another model that has a different vocabulary size of M, we can train decoding layers that can convert the output tokens of the generative model, into corresponding tokens from the discriminative model. However, this conversion layer would need to be trained beforehand, and may need to be frozen during the adversarial training, otherwise it would be a disconnect between the two models, and the input given to the discriminator may be corrupted. To train this conversion layer beforehand, one could use as a ground truth, the results that a tokenizer from model B, with the vocabulary size of M, would use as the targets in a cross entropy loss, and the results that a tokenizer from model A, with the vocabulary size of V, uses as the input to the layer.

4.5.5 Factuality in the Discriminator

Given that we can now adversarially train our models, we explore enhancing the discriminator with some way to determine factuality. We take a simple approach that in addition to the normal discriminator training objective (i.e., the discriminator is given a batch of generated text and a batch of real text and evaluated whether it inferred this correctly), we add a confounder loss. Our additional confounder loss is based on the confounder loss by [85], in that we shuffle around the subjects and objects and expect our model to determine that when shuffled objects are false. Since our generated outputs are structured (i.e., we have symbol tokens that delimit the different parts of assertions), we can do this shuffling easily and in-batch. Although shuffling may incur some false negatives (we may have a shuffled configuration that is factually correct), since we supply the story and target sentence, we expect the discriminator to be able to discern this correctly. We believe that we could also apply the max-margin loss utilized to great effectiveness by other language GAN literature [132, 31], although we leave this for future work.

4.5.6 Experimental Setup

For our GAN experiments, we had the following setup. We built a joint inference dataset using the procedure given in section 4.4, and we augmented it with hinting. Hinting was done in the same manner as in Section 4.3.1, by sampling a binomial distribution ($p = 0.5$) and if the sample is true we provide a hint by randomly sampling parts of the target assertion. This dataset is composed of 1,479,811 training examples, and 30,183 testing examples. We fed this data to a model with the adversarial setup described in the previous sections. The generator model utilized was the BART-base for conditional generation, and the discriminator model was a BART-base model for sequence classification.

For the sake of time, we run our model on only 100,000 examples, with a batch size of 32, on 3xNVIDIA A6000 for 3 epochs. In addition to this, to see the effects of the adversarial formulation and of the confounder loss, we train a model without the adversarial approach that we propose (a separated Generator and Discriminator with the confounder loss), and an adversarial model without the confounder loss to be able to gauge the effects of it. We train 4 random seeds for each of these 3 conditions. Additionally, to test the performance of the discriminator, we used the alignment technique from Section 4.4 to align the ConceptNet test set of [85] to the ROCStories corpus. We then passed the story, target sentence, and the test assertion to the discriminator to determine whether it was true or not. We used a threshold of 0.5 to determine whether an assertion was marked as true (1) or false (0).

4.5.7 Effects of adversarial training

After running automated tests, we see some mixed results between the three conditions (Adversarial+Confounder, -Adversarial+Confounder, Adversarial-Confounder). These results are in Table 4.9. We can see that the Adversarial models tend to have

Model	ROUGE1	ROUGE2	ROUGEL	ROUGELSUM	BLEU	METEOR	Accuracy of Discriminator
+ADVERSARIAL+CONFOUNDER	43.656	**10.544**	40.380	40.379	31.335	61.683	0.690
+ADVERSARIAL-CONFOUNDER	**43.747**	10.559	**40.530**	**40.531**	31.279	61.623	0.481
-ADVERSARIAL+CONFOUNDER	43.715	**10.680**	40.292	40.292	**31.470**	**61.776**	**0.794**

Table 4.9: We present the results of the adversarial test with ablations. We can see that the Adversarial models tend to have improved recall (ROUGE) scores, but lower precision (BLEU/METEOR) scores and lower accuracy on classifying a ConceptNet test set of assertions. The adversarial model with everything (+ADVERSARIAL+CONFOUNDER) strikes a balance of the benefits of precision and recall that the non-adversarial, and non-confounder loss models give respectively.

improved recall (ROUGE) scores, but lower precision (BLEU/METEOR) scores and lower accuracy on classifying a ConceptNet test set of assertions. The adversarial model with everything (+ADVERSARIAL+CONFOUNDER) strikes a balance of the benefits of precision and accuracy, and recall that the non-adversarial, and non-confounder loss models give respectively. Some possible causes for these mixed results may be that our approach may be too naive, and possibly an improved GAN formulation such as the Wasserstein GAN [7] used in [136] may help our results, our approximation to connect the generator and discriminator may be too naive and may need a more complex approach such as utilizing a recurrent neural network during the generation steps to encode them then decode them into the discriminator.

4.6 Contextual Commonsense Inference Summary

These systems (namely the "hinting", "joint-inference", and the adversarial/joint model) are novel contributions that are part of this thesis. Prior work was unable to control commonsense inference ([50], [113], [65]). This made utilizing these systems for specific purposes complicated or impossible, as it would need to be coerced in some other manner to be able to perform contextual commonsense inference about a specific topic/entity. Additionally, prior work trained one model per knowledge base, which can hinder inferences that utilize multiple knowledge bases, and requires one model per source which may require vast amounts of memory and compute capability to do inference. The work we have presented in joint inference is an innovative way to address these problems and works for different types of contexts and knowledge bases. Lastly, prior work did not have a unified method to score contextual commonsense assertions from multiple sources, which can lead to different scales in scores and makes it complicated to prioritize one fact over another. Our adversarial/joint model addresses this last point, through the Discriminator that learns to score contextual assertions as the generative model trains or separately by giving it context and a fact to score. Altogether, these systems push the boundaries in contextual commonsense inference and serve as a text-to-graph system. We use these as a text-to-graph engine to power the procedural abstraction that we present in the next section.

4.7 Procedure Ingestion and Abstraction System

So far, we have constructed a system to perform text-to-graph of a certain story context. For our particular use case, the stories that we will analyze and abstract into graphs will be procedures. This is important because we want to be able to utilize these contextual procedure knowledge graphs to generate procedures. However, we need a corpus of knowledge graphs from procedures first, to be able to train our step generation models and our reasoners. We now describe our approach to collecting this dataset and extracting graphs from it. We also describe our general approach to ingesting a procedure and converting it into a knowledge graph.

4.7.1 Procedural Abstraction Overview

The overall process to abstract a procedure into a knowledge graph is the following. First, we need to find a document that matches a certain procedure that we want to abstract. Following this, we need to extract the steps from this document. With the set of steps, we can then pass them as a story into our contextual commonsense inference system and generate assertions for the procedure. We utilize our hinting mechanism to be able to drive this inference process. The resulting assertions represent the facts that are necessary to perform a procedure, however these facts are not ordered, which poses an issue, because some facts may be more relevant/useful than others. We tackle this issue in the next Chapter. All along in this process, we can store the documents we have found, and we can store the facts that we have generated for easier retrieval later. The general process can be visualized in Figure 4-7.

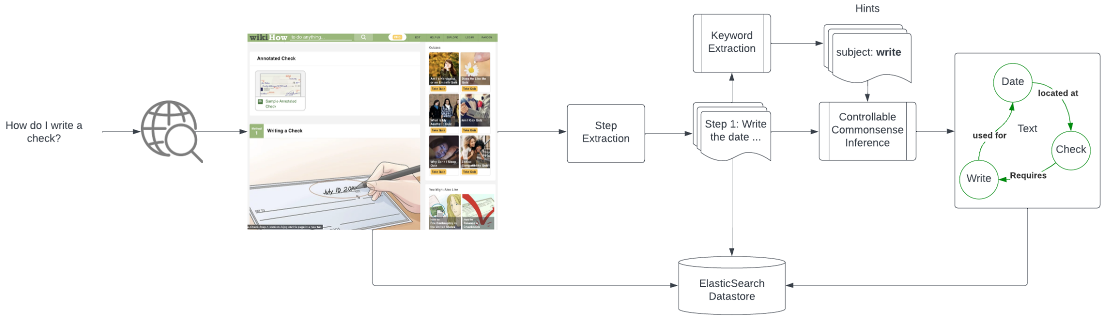


Figure 4-7: Process to ingest and abstract a procedure. We go from a procedure title, to a document that contains the procedure, to steps that are extracted from the procedure, and finally into facts that are generated from a hinted controllable commonsense inference system. The original data, the steps, and the extracted commonsense assertions are all stored in ElasticSearch.

4.7.2 wikiHow as a Procedure Dataset

For our step generation model, and our baseline gold instruction model later on, we require a large corpus of procedures to be able to train and develop them. The simplest way to acquire this, is by scraping the wikiHow website. wikiHow is a website that provides how-to guides on a wide range of topics. It was created in 2005 as a platform where people could share their knowledge with others. The site has grown to become one of the largest how-to websites in the world, with over 200,000 articles covering topics such as cooking, gardening, home improvement, and more[8]. wikiHow is particularly useful, because the structure of procedures are in a step-by-step manner. In addition to this, there is what can be seen as a summary step, and a longer detailed step. There is an existing wikiHow dataset [76], however this dataset was focused on summarization. The authors focus on utilizing the detailed steps as an input to a summarization model, and producing the summarized steps. There are also requirements to do the procedure, warnings, and references to validate the procedure.

---

[8] `https://www.wikiHow.com/`

In our work, we require the structure of the procedure: we need the order of the steps and any additional information that may have been provided in the procedure (such as the requirements and warnings). We need the steps for our step generation model presented in the next chapter, and we need the additional knowledge, because we utilize this as an additional knowledge base for our contextual commonsense inference model, which we describe further below. Because of this requirement, we opted to scrape our own version of wikiHow.

Scraping wikiHow

To be able to scrape wikiHow, we utilized the scrapy Python package [9]. Through scrapy, we implement a Spider that utilizes the Beautifulsoup 4[10] package to parse the HTML from wikiHow into usable fields. The scrapy package handles following links into related pages, along with any timeouts and retries that might be encountered while scraping. We note that wikiHow provides a Creative Commons license for their articles[11]. Ultimately, from pages that we scrape, we extract the items found in Table 4.10. We note that in the Table, we additionally include fields that we utilize for other purposes when storing the data in an ElasticSearch instance.

We additionally save the HTML content of the article for future endeavors that may utilize it. In our effort, we let our crawler run for one week. In that week, the crawler was able to collect 93144 articles. The articles contain on average 2.77 sub-procedures, and contain on average 5.15 steps per sub-procedure. There are a total of 258230 sub-procedures, and 1331020 steps in our dataset. Sub-procedures in this case can be seen as and/or trees to accomplish a higher level goal (procedure). The "and" trees, or set of steps that are required to accomplish a goal, in this case are sub-procedures that wikiHow calls "parts", and the "or" trees, or alternate (possibly not required) set of steps that represent different ways of accomplishing a goal, are what wikiHow calls "methods".

---

[9] https://scrapy.org/

[10] `https://pypi.org/project/beautifulsoup4/`

[11] https://www.wikiHow.com/wikiHow:Terms-of-Use

Field	Description	Example
Content	Field that is utilized to generate an embedding for fast nearest neighbors search of procedures. Consists of the title of the procedure.	How to Connect Your PC to Your TV Wirelessly
Embedding	Field that stores an embedding that represents the encoded value of the Content field	[0.12, 0.123, 0.213 [...], 0.0102]
Gold Knowledge Graph Completed	Field that indicates if there is a corresponding knowledge graph generated for the procedure	True
Ingredients	Field that contains any ingredients (in the case of cooking procedures)	["lettuce", "salt"]
Intro	Field that contains any introduction or preamble to the procedure. Usually there is a brief summary of the procedure at hand.	Connecting your PC to your TV can help when you want to cast videos such as sporting events or playing video games.
KDPSG Cache	Field that contains steps generated from the Knowledge Driven Procedural Step Generation system. These are cached for speed.	[["Step 1: Open Settings", "Step: ..."]]
References	In the case of a procedure, any references that the procedure may contain	Luigi Oppido. Computer & Tech Specialist. Expert Interview. 31 July 2019
sub-procedures - Steps - Detailed step	Longer more detailed description of a procedural step	If your TV is an Android TV, is powered by Chromecast, or has a Chromecast device attached, you can mirror any Chromecast-enabled Windows apps to the screen. Many Windows apps support Chromecast, including Google Chrome, Netflix, Spotify, and Plex. If you have the Google Chrome web browser, you can use it to mirror any website to the TV. This includes websites like YouTube, Netflix, and Facebook. Google Chrome also lets you mirror your Windows PC's entire screen!
sub-procedures - Steps - Step Num	The number of the step in a sub-procedure.	1
sub-procedures - Steps - Summary Step	A sentence that summarizes a step in a procedure. These can be seen as lower level goals in a procedure.	Turn on your Chromecast-enabled TV.
sub-procedures - Title	The title for a sub-procedure	Using Chromecast
sub-procedures - Type	Whether a given sub-procedure is a part (set of sequence of steps) that needs to be done in order or a method (alternate way of doing something). Methods can be seen as OR nodes in goal trees and Parts can be seen as AND nodes. [FIND REFERENCE]	Methods
Summary	A summary of the steps that need to be done in the procedure	1. Turn on your Smart TV. 2. Connect your PC to the same Wi-Fi network as the TV. 3. Open Google Chrome. 4. Click the three-dot menu and select Cast. 5. Choose what you want to cast. 6. Click your TV.
Things Needed	Any requirements that may be needed apart from ingredients	Chromecast enabled TV, PC
Title	The title of a procedure. This is the higher level goal that is to be achieved by the procedure.	How to Connect Your PC to Your TV Wirelessly
Warnings	Any warnings that the procedure may have	Be careful with flashing lights if a person is epileptic.

Table 4.10: Table that describes extracted fields from the scraper.

Storing wikiHow

To be able to store and search through our procedures, we utilize ElasticSearch. It provides a simple and efficient platform for ingesting and indexing large amounts of data. We have 4 separate indexes in ElasticSearch: an index for generated procedural knowledge, an index for registered users for our conversational agent, an index for user interactions with the conversational agent, and an index for ingested procedures. Indexes can roughly be seen as tables in a SQL database; an index is a collection of documents that have somewhat similar characteristics. We store our scraped and parsed wikiHow dataset in this last index. Later on, the facts that we generate from our text-to-graph systems are stored in the first index mentioned. The other two indices are for the conversational system that we present in Chapter 6. We now give a description of the indices and the data that they hold:

- **Ingested Procedure Index** In this index, we store any procedure that we may have ingested through our systems. We use the following fields: content, embedding, ingredients, intro, references, sub-procedures - steps - detailed step, sub-procedures - steps - step num, sub-procedures - steps - summary step sub-procedures - title, sub-procedures - type, summary, things needed, title, warnings. These are based on the aforementioned scraping. The description of these fields can be found in Table 4.10.

- **Contextual Procedure Knowledge Index** In this index, we store any fact that is generated from the Contextual Commonsense Inference model applied to a procedure. The fields that we save are the following: discriminator score, generation score, content, general, object, relation, sentence, step, story, subject, sub-procedure. We describe these fields in Table 4.11. We opted to store these assertions in a separate index, because in earlier tests, if stored within a field in a procedure, the ElasticSearch indexing would eventually crash because of too many sub-fields and variations.

Field	Description	Example
Discriminator Score	Score given by discriminator described in Section 4.5	0.900
Generation Score	Generated sequence score given by the generator model described in Section \ref{}	-0.272
Content	Searchable field which contains the name of the procedure that the fact was extracted from, and a textual form of the assertion	Procedure: How to do screen mirroring and projecting to your PC Step: 1 Fact: Specifically, mirroring SimilarTo projection.
General	The specificity of the assertion as defined in Section \ref{}, in the form of a boolean confirming if the assertion is general or not	False
Object	The object of the assertion	projection
Relation	The relation that joins the subject and object node. We store it as it is found in other KGs.	SimilarTo
Sentence	The sentence from which the fact was derived	Step 1, Select Start>Settings>System>Projecting to this PC
Step	In the case of a procedure, the step from which a fact was derived from.	step_1
Story	The story/procedure from which the fact was generated from	How to do screen mirroring and projecting to your PC: Windows 10. Step 1, Select Start>Settings>System>Projecting to this PC
Subject	The subject of the assertion	mirroring
Sub-procedure	In the case of a procedure, the index of the sub-procedure from which the assertion was generated	0
Document Index	The id of the document in the Ingested Procedure Document index	oeQUGocBi6St9mxkQhTf

Table 4.11: Fields and their descriptions for the index that stores contextual procedure facts.

Retrieving procedures from wikiHow

Once the procedures are in ElasticSearch, we utilize the haystack library [131] to implement a semantic search that utilizes the sentence-transformer [146] library. Haystack additionally utilizes the FAISS library [71] to quickly index and perform fast approximate nearest neighbors on the knowledge. With this, we can additionally retrieve procedures for training our step generation models. The haystack library generates a sentence-transformer embedding for a “content” field. In our work, for procedure documents, the “content’ field is the title of the procedure, and for contextual procedure assertions, the field consists of the textual form of the assertion (simple English description of the tuple e.g. a dog, isA, animal is described as A dog is a animal.) along with the step and the procedure that it was extracted from. For future work, we note that this storage strategy could be utilized to search for facts that may have been used in similar procedures.

4.7.3 When retrieval doesn’t match: dynamic lookup and extraction

We tested the haystack-based procedural retrieval system, and found that items with a cosine similarity of less than 0.9, tended to be procedures that are not in our dataset. This is an issue later on when utilizing conversational agents, because a user may ask them for a specific how-to and if we rely purely on the most similar item to a user’s query, we may retrieve an irrelevant procedure. An example of this would be with the query: "How to organize applications on an iPhone". From our collection of procedures, the top procedure is "How to Organize iPhone applications into folders" with a cosine similarity of 0.945. However, if this procedure were not available, the next most similar procedure is: “How to Move Apps on the iPhone”, with a score of 0.86 which is similar but not exactly the same as organizing apps in folders. We tried out several other procedures, and found that typically the desired procedure had a similarity of higher than 0.9. In future work, this however may not be an issue directly: if we had a system that retrieved facts based on an input query, it would be able to come up with the steps for a procedure.

Now, we address the issue of what if the similarity is lower than this, and we do not have the procedure available? This could also be seen in future work, as if the similarity of the facts to the query for a procedure was on average below a certain threshold. This question is key to our approach, because ideally we would want an agent that can handle procedures that are **not** in our collection of knowledge.

The way we do this is by performing a web-search for the procedure, using a span extraction model to extract steps, and storing the procedure. In Future Work, an even more ideal way to handle the procedure generation process would be to utilize the semantic search for assertions in combination with a reasoner to be able to not have to rely on a search engine nor on a span extraction model. The semantic search retrieves whatever facts it deems necessary for the procedure, and the reasoner orders them into something useful. For this work, we utilize an unofficial wrapper around the Google search engine[12], however any other web search would work well, and input the user's query (or unknown procedure) along with a prompt: "How to [DESIRED PROCEDURE]". We then take the first of the results and proceed to do the following. We note that it might be possible to do this for all the results that are retrieved in parallel to have as much coverage as possible.

To be able to ingest the procedure, and it is useful further down in our pipeline, we need to have, at a minimum, the title of the procedure, and the summary steps. However, in websites, procedures are sometimes spread out through the page, and formatting is inconsistent. What this means is that there may be a procedure contained in an ordered list (<*ol*>) element in one page, in another the procedure may be in explicit text with numbers (e.g., 1. [STEP 1 TEXT]), and in a third page, the procedure might be in an unordered list (<ul>) element. Therefore, it is important to have a viable way of extracting or delineating where procedure steps are.

To perform some initial filtering and standardization, we run the resulting page through a programmatic readability system[13] to try to remove most of the junk/ads that may be present. Additionally, this standardizes lists (both ordered and unordered) into asterisk lists (*). Now, prior work has used models to find spans of text that represent individual steps [87]. While this is a viable approach, we would need a much larger annotated dataset to be able to train a span extraction model on general web-based procedures.

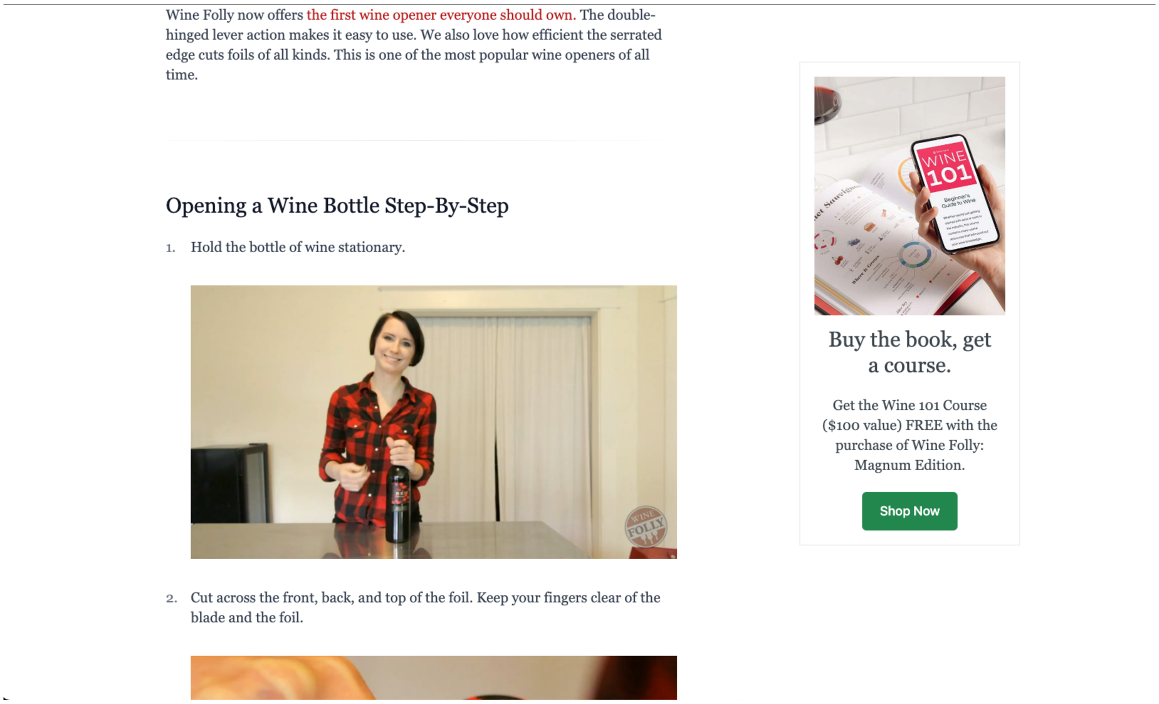
This inspired us to utilize a LLM to do step extraction. We note that this LLM

---

[12] https://github.com/abenassi/Google-Search-API

[13] We use the "readabilipy" library: https://github.com/alan-turing-institute/ReadabiliPy

can be seen as a module that can be replaced with another model or sub-system that ingests text and produces steps found in the text. We used the LLM simply for convenience. The input only needs to be a web page's text, and the output a list of steps. Ultimately, whatever system is used in this stage, one will have a list of steps at the end. This list of steps can be utilized as a baseline for how to do the procedure, and in our case, we can use it to convert it to a contextual procedure knowledge graph which we describe in the next section. We now show how this process would look like for the procedure of "How to open a wine bottle", taken from the website: `https://winefolly.com/episode/how-to-open-a-bottle-of-wine/`, found by a Google search for the procedure. This can be seen in Figure 4-8.



Wine Folly now offers the first wine opener everyone should own. The double-hinged lever action makes it easy to use. We also love how efficient the serrated edge cuts foils of all kinds. This is one of the most popular wine openers of all time.

Opening a Wine Bottle Step-By-Step

1. Hold the bottle of wine stationary.

2. Cut across the front, back, and top of the foil. Keep your fingers clear of the blade and the foil.

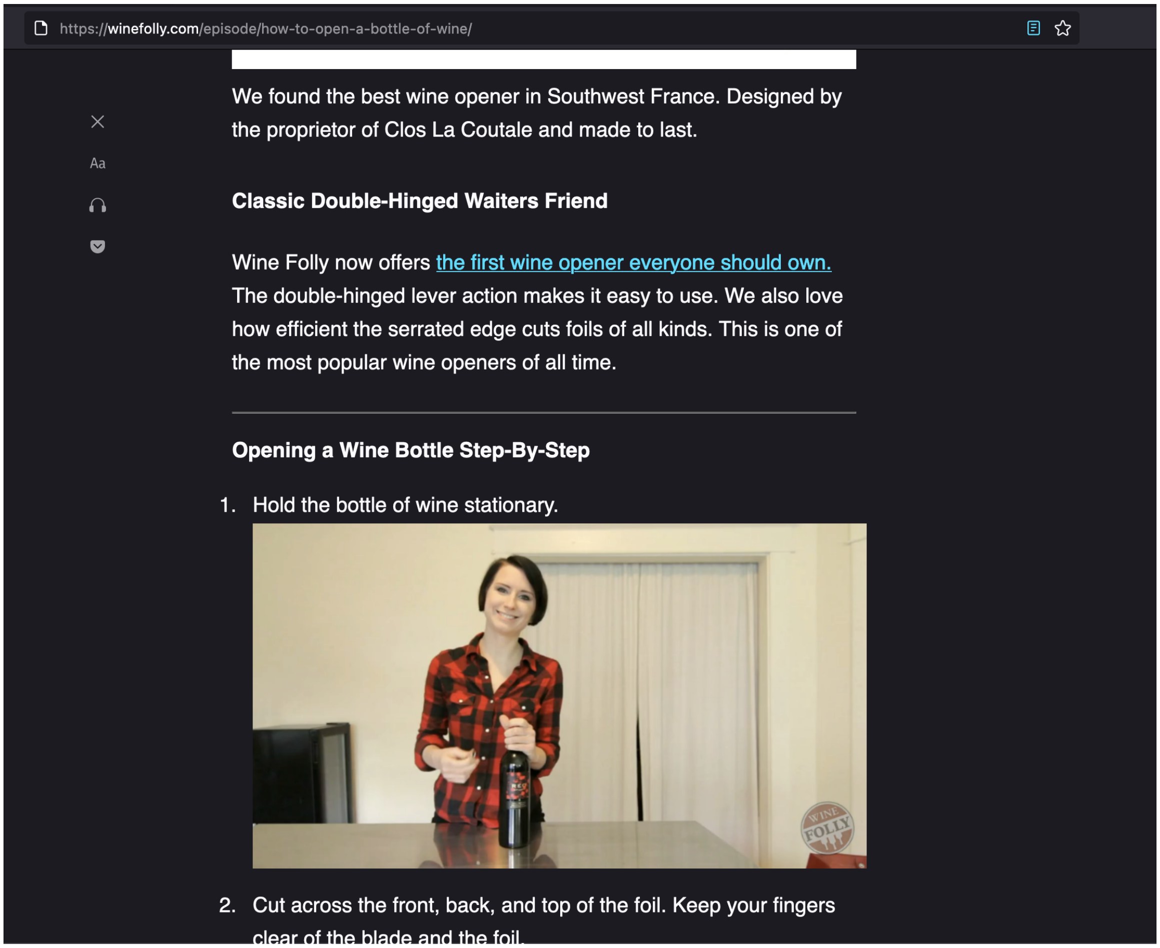
Buy the book, get a course.

Get the Wine 101 Course ($100 value) FREE with the purchase of Wine Folly: Magnum Edition.

Figure 4-8: Web page view for wine opening procedure. Note that above it there is more text and below there are more steps and images.

Following this, we apply readability to the page. To visualize the effect of this, we use the Firefox readability view, which is an equivalent of the "readabilipy" library that we utilize. The resulting text can be seen in Figure 4-9.

After this, we take the textual content of the simplified page, and pass it to our LLM. In our case, we utilize the GPT-3, "text-davinci-003 model". We give the model



We found the best wine opener in Southwest France. Designed by the proprietor of Clos La Coutale and made to last.

Classic Double-Hinged Waiters Friend

Wine Folly now offers the first wine opener everyone should own. The double-hinged lever action makes it easy to use. We also love how efficient the serrated edge cuts foils of all kinds. This is one of the most popular wine openers of all time.

Opening a Wine Bottle Step-By-Step

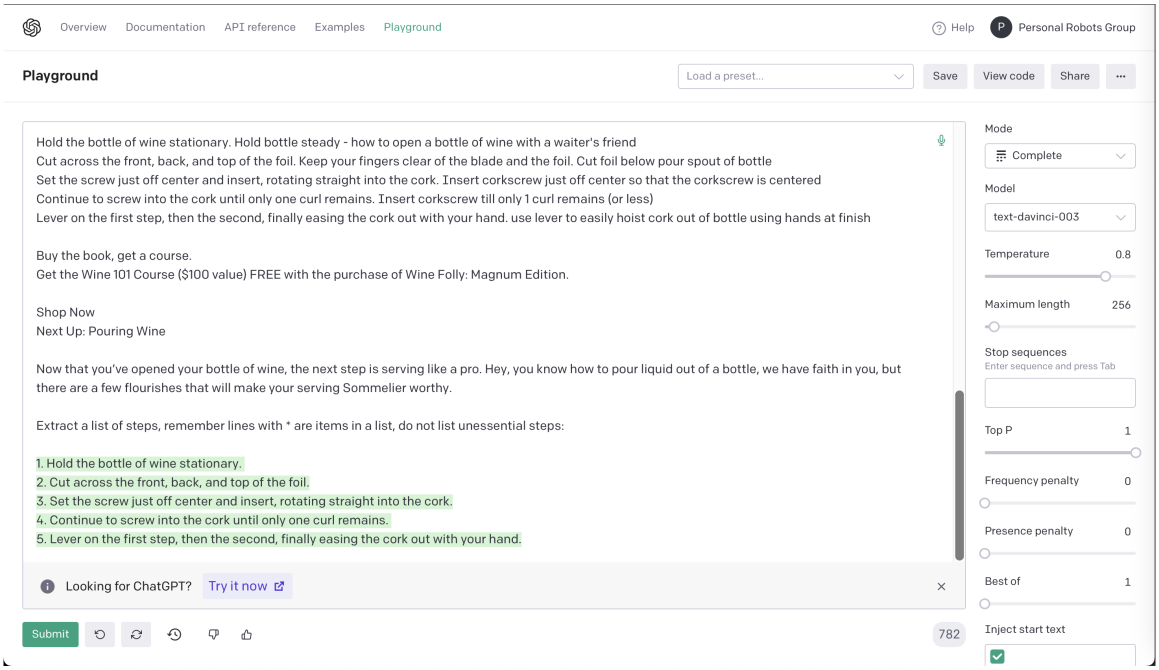
1. Hold the bottle of wine stationary.
2. Cut across the front, back, and top of the foil. Keep your fingers clear of the blade and the foil.

Figure 4-9: Readability view for wine opening procedure. Note that the advertisements from above have been removed, however there is still extra text above the procedure.

the following prompt to perform the task: "For the procedure:[PROCEDURE NAME] Described by: [READABILITY CONTENT] Extract a list of steps, remember lines with * are items in a list, do not list unessential steps:". We see an example of how this works, we show the OpenAI Playground in Figure 4-10.

Ultimately, the LLM produces the following list of steps:

1. *Hold the bottle of wine stationary.*
2. *Cut across the front, back, and top of the foil.*
3. *Set the screw just off center and insert, rotating straight into the cork.*
4. *Continue to screw into the cork until only one curl remains.*



Hold the bottle of wine stationary. Hold bottle steady - how to open a bottle of wine with a waiter's friend
Cut across the front, back, and top of the foil. Keep your fingers clear of the blade and the foil. Cut foil below pour spout of bottle
Set the screw just off center and insert, rotating straight into the cork. Insert corkscrew just off center so that the corkscrew is centered
Continue to screw into the cork until only one curl remains. Insert corkscrew till only 1 curl remains (or less)
Lever on the first step, then the second, finally easing the cork out with your hand. use lever to easily hoist cork out of bottle using hands at finish

Buy the book, get a course.
Get the Wine 101 Course ($100 value) FREE with the purchase of Wine Folly: Magnum Edition.

Shop Now
Next Up: Pouring Wine

Now that you've opened your bottle of wine, the next step is serving like a pro. Hey, you know how to pour liquid out of a bottle, we have faith in you, but there are a few flourishes that will make your serving Sommelier worthy.

Extract a list of steps, remember lines with * are items in a list, do not list unessential steps:

1. Hold the bottle of wine stationary.
2. Cut across the front, back, and top of the foil.
3. Set the screw just off center and insert, rotating straight into the cork.
4. Continue to screw into the cork until only one curl remains.
5. Lever on the first step, then the second, finally easing the cork out with your hand.

Mode: Complete
Model: text-davinci-003
Temperature: 0.8
Maximum length: 256
Stop sequences: Enter sequence and press Tab
Top P: 1
Frequency penalty: 0
Presence penalty: 0
Best of: 1
Inject start text: ☑

Figure 4-10: Visualization of a LLM being utilized for step extraction. Note that the LLM removed unessential text from the procedure, and ignored useless text.

5. *Lever on the first step, then the second, finally easing the cork out with your hand.*

We can then parse these steps and use them as summary steps to ingest into our data-store. We note that although we showed this in the OpenAI Playground, that there is a programmatic equivalent with the "openai" Python package.

With this strategy, whenever a procedure that is not exactly or close enough to what a user wants, we can extract a way to do it from the web on the fly.

We note that some behaviors that we observed in this process, were that it may be the case that the web-page has more than one procedure. If this is the case, sometimes the large language model may intermix procedures, it may also ignore procedures. Because of this, other more specifically trained procedure step span extraction methods may be preferable. However, we find that this method is effective as a proof-of-concept.

4.7.4 Contextual Procedural Knowledge Generation

With our wikiHow dataset, and our method to generate procedure documents, we now focus on how to convert these texts into contextual knowledge graphs.

To convert our procedural steps into knowledge graphs, we do the following procedure. For every step in a procedure (assuming that it is a single sentence), we extract the noun and verb phrases of the step utilizing the spaCy [60] library. Following this, we use these noun and verb-phrases as *subjects* in permutations with the relations from ATOMIC-2020 for hinting. In essence, for every step in a procedure our contextual commonsense inference system generates a specific assertion without any hint, and with hints consisting of a noun/verb phrase as a subject, and a hint consisting of permutations of noun/verb phrases and ATOMIC-2020 relations.

We can visualize this in Figure 4-11.

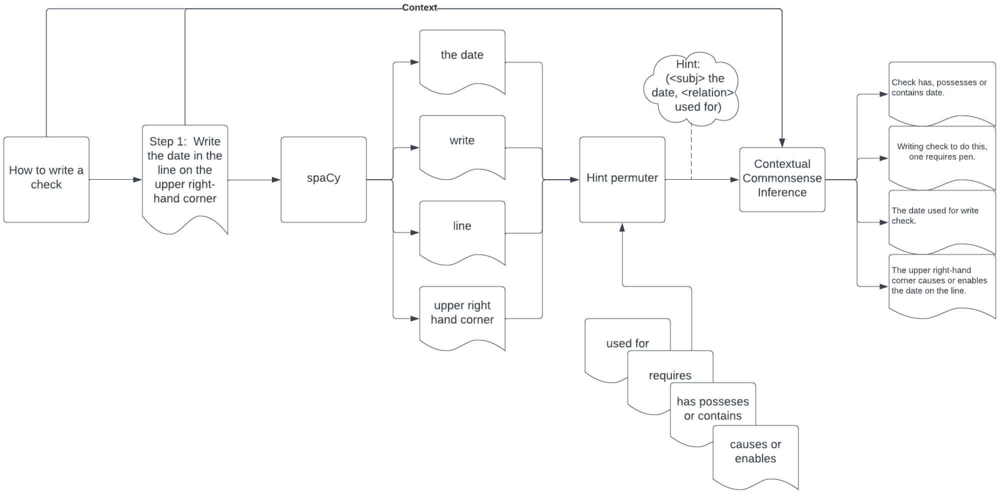

Context
How to write a check
Step 1: Write the date in the line on the upper right-hand corner
spaCy
the date
write
line
upper right hand corner
Hint permuter
Hint: (<subj> the date, <relation> used for)
Contextual Commonsense Inference
Check has, possesses or contains date.
Writing check to do this, one requires pen.
The date used for write check.
The upper right-hand corner causes or enables the date on the line.
used for
requires
has posseses or contains
causes or enables

Figure 4-11: Procedure for abstracting Step 1 in the procedure how to write a check. We can see the extracted noun/verb phrases and how these are used in permutations with relations to produce hints to guide the generation. We note that this is a small example of the contextual knowledge graph for Step 1, and that there are more assertions that are generated from the permutations.

We note that although we can control the contextual commonsense inference process, even for a small amount of steps, it is possible to generate thousands of assertions. Before we perform the optimization found in Section 5.3.1 (grounding the inference to be about noun/verb phrases on a target sentence), we were utilizing keywords and noun/verb phrases from the complete procedure, along with relation-object and object hints. The process for generation that we use for most experiments until Section 5.3.1, although more comprehensive in the knowledge that it generates, generated thousands of facts per step. Which, for a language model with a limited context, raised the challenge of which facts can we utilize to optimize our step generation. As we generate facts, we score the generated facts for their plausibility given the context with the discriminator that we describe in Section 4.5. When we combine the generation with the scoring system. we can construct a knowledge graph that is grounded on entities (noun/verb phrases), and has a metric of the strength of the assertion.

Performance Optimizations

The generation procedure for assertions is recurrent, what this means is that for a large scale generation of facts the process can be computationally slow. If we add to this, the scoring in the discriminator, what happens is that the whole process becomes even slower. To remedy this, we opted to use ONNX-runtime[14] along with the huggingface Optimum library[15] to try and optimize our models for generation. ONNX runtime provides a way to perform compute level graph, optimizations, which when performing inference in a model speeds it up. We found that this speed up was very effective for the discriminator, to the point that after optimizing it through ONNX, it can compute scores for a batch of 32 items on average at ~ 0.2 seconds. However, we found that the gains were marginal for the generator (~ 5 seconds for 32 items vs. ~ 7 seconds), and the quality of the results of the generated text seemed to be inferior to the original model. Since graph-level optimizations did not work all that well on the generator, we opted to parallelize the process. One optimization that we did, was to do a multithreaded inference in which four threads would process a queue of contextual common sense inference requests, and this way we could parallelize and speed up the generation process. The times that we noted above for inference are for the multithreaded system, we used four GPUs in parallel. The original compute time

---

[14] https://onnx.ai/

[15] https://huggingface.co/docs/optimum/index

was approximately 4X the times that we stated. We note that in much of this work, we utilize Nvidia's A6000 or A40 GPUs.

We note that by taking the optimizations, and abstracting process shown in Figure 4-7 and in more detail in Figure 4-11, we can build a function that can take the name of a procedure and output a contextual knowledge graph for it in a relatively small ($\sim$2 minutes per procedure).

4.8 Summary

In this fourth chapter, we describe the systems we developed to perform contextual, common sense inference for procedures. These systems utilize large language models, specifically transformer models, to ingest a story or procedure and goal. We then generate a knowledge graph for each sentence or step of the procedure, incorporating multiple knowledge sources and enabling us to score generated assertions. A unique contribution of our system is that it can be controlled, which allows us to fine-tune its performance for specific tasks. We also provide a proof-of-concept system that can extract procedural steps from web searches for a specific procedure. To train our step generation model and apply our reasoner, we contribute a more detailed wikiHow dataset. Our work addresses the problem of providing contextual, common sense inference for procedures, and our contributions enable more accurate and effective procedural guidance.

Chapter 5

Processing Abstracted Procedural Knowledge into Steps

Life doesn't come with an instruction manual. -Scott Westerfeld

5.1 Overview

By now, we have discussed how we can generate knowledge graphs that represent procedures. However, what can we do with those graphs? Here, we discuss utilizing portions of them to generate procedural steps. We look at the need for knowledge in the task of procedural step generation, and look at different ways of trying to organize knowledge into something that is effective for step generation. The overall process consists of selecting a certain set of facts and converting it into text that is a step in a procedure. As we will show, the order and the content of these facts matter greatly. This task can also be seen as data-to-text or graph-to-text, with the added constraint that it needs to be coherent with past steps, and needs to be accurate for completion. We show that it is indeed possible to do this, and that this can be done to generate steps for out-of-domain procedures.

5.2 Knowledge Driven Procedural Step Generation

5.2.1 Problem Statement

The task that we are trying to tackle can be formally called graph-to-text generation, a subtask of data-to-text generation [52]. We have a knowledge graph that represents a step in a procedure, and we need some system that can convert that knowledge graph into text that would be seen as the procedural step. However, we have two problems that we need to address in this task. The first one is selecting the correct graph to input into the system, and the second one is actually the translation of that graph into text. Formally, we utilize the notation from [183]. Our task consists of given a source-target data pair X, Y, we aim to generate a procedure step sentence Y from the input X. The sentence Y, or the target text, can be decomposed in such a way that $Y = y_1, y_2, ..., y_m$ in which y is a token. The task can also be generalized to say that Y can also be represented as $Y = s_1, ..., s_k$, where s is a sentence including multiple tokens. We diverge from the definition given in [183], such that we define our input X source text, to have the following components. Firstly, a collection of facts that can be seen as $X_f = x_1, x_2, ..., x_n$ which denotes the input records. An element x represents a record that is a tuple including multiple features, i.e. $x = (r_1, ..., r_{|x|})$. Recall that in Chapter 4, we have developed a system that can generate a contextual knowledge graph of a procedure. The contextual knowledge graph contains assertions in the form of tuples, that can be mapped directly to this definition of x. X also contains the title of a procedure, x_{PT}, the step that we want to generate in the procedure $x_{SN} := \{Step\ 1, Step\ 2, ..., Step\ i\}$ (the step to be generated), and any steps prior to *Step i*: $x_{PS} = \emptyset, s_1, s_2, ..., s_{i-1}$. Now, the records that we include as inputs for our step generation model, can be ordered in a variety of ways. We call each ordering of these a plan, $\mathcal{P} \in X_f$. Now, all together, our task can be seen as: $Y = realizer(x_{PT}, x_{SN}, x_{PS}, \mathcal{P}) = realizer(X)$, where *realizer* is a generative text model that takes the given inputs and produces a procedural step Y. We call it the "realizer", because it realizes the plan that was given as the input.

We utilize this formulation, because there is a limit for the amount of information that can be contained in the plan $\mathcal{P}$, due to memory and compute constraints, a maximum amount of inputs that can be given as input to the realizer model. Additionally, the space of facts X_f may contain more facts than are actually needed for a step. Because of these considerations, we would need a planner system that can, given the all the facts and the appropriate context, generate a suitable $\mathcal{P}$. This can be defined as: $\mathcal{P} = planner(x_{PT}, x_{SN}, x_{PS}, X_f)$.

We note that we simplified the formulation of this task in such a way that we constrain X_f to be only the knowledge that was generated from a step. There is a more complex formulation in which X_f can be all the facts that stem from the procedure (from every step, and any other text that we include in the contextual commonsense inference). We additionally note that we are simplifying to only generate one step at a time. A more complicated scenario would also be to generate all the procedure steps at once. These formulations remain future work, and we leave all the data necessary to explore these.

5.2.2 Procedural Step Realization Model

Before exploring the planner aspect of this work, we first decide to implement the realization model. Our reasoning for this, is that we can train the model in a general capacity by making plans $\mathcal{P}$ by randomly sampling the step knowledge available. Then, once the model is trained to realize the plans, we can find a way to generate better plans. This approach of pre-training the model is one that has been utilized in other refinement approaches, such as for reinforcement learning [185]. As a realization model, we utilize a transformer encoder-decoder model. In our work, we utilize the transformer FlanT5[95] base model, as it was currently one of the highest performing base models in a variety of tasks. We utilize encoder-decoder systems, as these have been shown to have good performance in the task of graph-to-text generation [148].

5.2.3 Dataset

To be able to train our procedural step realization/generation model, we first need to generate a dataset of procedures aligned with their corresponding contextual knowledge graphs. We utilize our contextual commonsense inference model from Chapter 4 for this. We note that the model that we utilized, in addition to having ATOMIC-2020, ConceptNet, and GLUCOSE, it also has Ascent++ and facts that were extracted from the wikiHow dataset. These facts were: requirements and ingredients for a procedure, warnings for a procedure, along with has first/last sub-event facts. These were extracted from the scraped procedures, typically in the format of *(specific, [PROCEDURE NAME], [RELATION], [WARNING / REQUIREMENTS / INGREDIENTS / SUB-EVENT])*. We additionally used the summary steps in the procedures to serve as *is before/is after* assertions (i.e., [Step 1 content] is before [Step 2 content]). Lastly, we utilized the procedures that we scraped from wikiHow as an additional corpus of stories, where every story's first sentence is the name of the procedure, and every other sentence is the summary steps that were extracted. We did this so that our contextual commonsense inference would be able to generate assertions from stories longer than 5 sentences, and so that we could have commonsense knowledge applied to procedures.

With our contextual commonsense inference model, we then proceeded to go through the procedures that we had scraped, and perform contextual commonsense inference. Initially, we opted to generate as many inferences as possible. To do this, we would extract keywords/noun phrases/verb phrases from the complete procedure, and we would use these as subjects and objects in hints, with permutations of relation types. This process can be seen in Figure 5-1, and we describe the general approach in Section 4.7.4.

Excess of Assertions

While our strategy to generate assertions may be a comprehensive process, it is a very computationally intensive process, what this means is that it took approximately 5–10

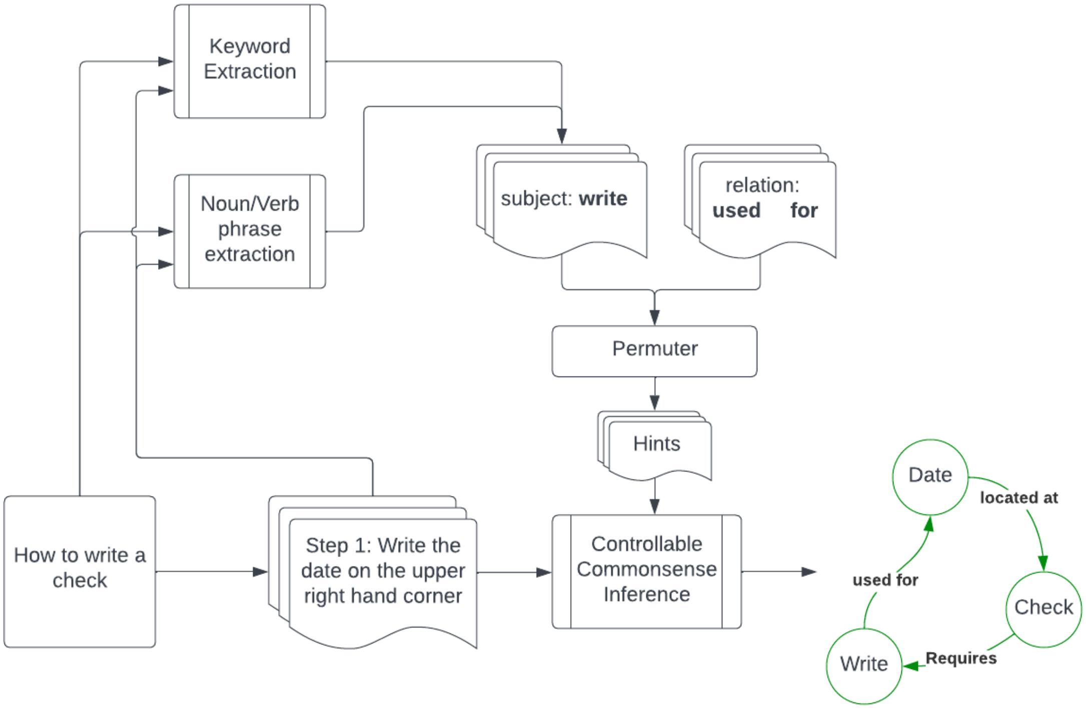


Figure 5-1: Visualization of process to generate contextual commonsense assertions. Note that the extraction systems receive *all* of the procedure's text including the title to generate keywords/phrases from, and this is what is utilized to do contextual commonsense inference.

minutes to produce a knowledge graph for 1 procedure. Because of this, we were only able to produce 37,172 of 93,144 procedures. However, these graphs contain upwards of 349 million assertions. There are close to 10,000 assertions for every procedure.

We found that this strategy of using procedure-level keywords/phrases produced too much information and had to apply some filtering to reduce this. We kept assertions whose rouge [90] scores between the assertion and the target step were above the average of all step related knowledge and a small delta (i.e., $rouge(assertion,\ procedure\ step) > \frac{1}{N}\sum_{n=1}^{N} rouge(assertion_n, procedurestep) + \delta$). We select $\delta = 0.04$. This strategy leaves us with assertions that have some overlap (and hopefully semantic relevance) to the procedure step. We utilize rouge as a way to try and enforce that the model is grounded in what it utilizes to generate steps. We note that other metrics such as BLEU[124], or BERT-Score[190] could be utilized for this.

	Validation						Test					
Condition	BLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM	BLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM
No Knowledge	23.327	54.022	46.738	26.203	45.652	45.667	23.157	53.701	45.978	25.942	45.194	45.238
Noisy Knowledge	33.995	62.098	57.861	40.413	56.961	56.885	34.596	62.592	59.103	42.213	58.173	58.194
Filtered Knowledge	49.565	75.994	77.256	63.733	75.213	75.217	51.646	77.616	79.143	65.807	76.976	76.995

Table 5.1: Validation and test results for training models for procedural step generation with and without knowledge.

This strategy filters out approximately 60% of facts from all of our assertions. However, this is not a viable strategy at inference time, because we would need the generated step or some equivalent of it to be able to calculate it. We discuss in the reasoner section later on our approach to handle this.

5.2.4 Experiments, Results, and Analysis

The need for contextual knowledge

The first thing that we set out to check, was whether knowledge the contextual commonsense assertions were necessary for the procedural step generation. To test this, we picked a subset of 1000 procedures, which were the most semantically related to the search term "software, smartphones, apps". We picked this so that the subset would be as similar as possible and would hopefully generalize well. With this subset of 1000 procedures, we trained the realizer model from scratch *without* any contextual knowledge, then we trained one with 32 facts that were randomly sampled without any rouge filtering, and finally we trained one with 32 facts that were randomly sampled from the rouge filtered knowledge. The results of this experiment can be seen in Table 5.1 and in Figures 5-2. We ran 3 seeds for every one of these test cases for 3 epochs. Additionally, we utilized a batch size of 8 and a learning rate of 4e-5 with a linear decay with 20% of the batches as warm up. Additionally, we use a train/dev/test split of 80/10/10.

We note that we remove duplicate knowledge to try and minimize, albeit slightly, the amount of information that the model has to process, and we also remove any assertion that is very similar (cosine similarity higher than 0.7 with respect to the target procedural step Y). To remove the duplicates, we go through the assertions that we have for a given step, and find any following assertion whose cosine similarity

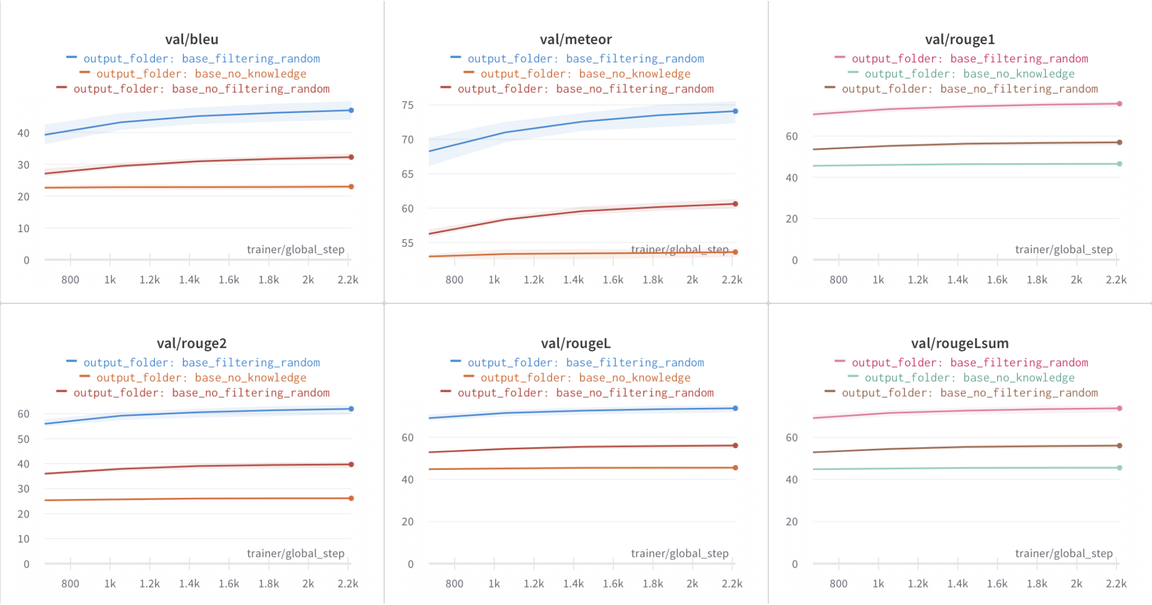


Figure 5-2: Validation results for tests of whether knowledge was present or not. We can see that without knowledge, the model's performance stays almost constant on automated metrics. However, we see that with some noisy knowledge, the performance of the model slowly improves. We see that with filtered (i.e. more relevant/grounded) knowledge, the model performs considerably better.

is higher than 0.98 of a given assertion.

From these experiments, one thing is very clear. There is a definite need for knowledge; without any knowledge, the system underperforms. Even with some noisy knowledge, we can see that the performance jumps considerably. We can also see that if we have higher quality, or more relevant knowledge, that the performance goes up even further. Now, this raises the question of what is an adequate amount of knowledge for this task.

How much knowledge?

Now that we know that we need knowledge to be able to do procedural step generation, we now try various values of the amount of knowledge to determine what is the sweet spot for the maximum length of our plan. We evaluate the following values of j in $|\mathcal{P}| = j; j = \{1, 2, 4, 8, 16, 32, 48, 64, 128\}$. We note that values higher than 64 tend to be truncated (the T5 model that we use has a maximum context size of 1024 tokens, which, if we take an assertion to be a sentence, and a sentence has roughly 10-20+

tokens, in an optimistic setting where assertions are relatively short, we have filled more than half of the context with the facts. If we include the procedure name, the step that we want to generate, and prior steps, it makes it almost impossible to be able to consistently include more than 64 facts. We include 128 to see what benefits arise from including facts until truncation. In contrast to the prior set of experiments, we only utilize the case with the random sampling from the rouge-filtered facts. We do this so that the quality of the facts is the highest, so we can see what the quantity does.

Once more, we perform our tests on a subset of 1000 procedures that are the most similar to the query "smartphones, software, apps". Additionally, we once again use a similar setup: we ran 2 seeds for every one of these test cases for 3 epochs. Additionally, we utilized a batch size of 8 and a learning rate of 4e-5 with a linear decay with 20% of the batches as warm up. Additionally, we use a train/dev/test split of 80/10/10. The results of these experiments can be seen in Table 5.2 and in Figure 5-3 and 5-4.

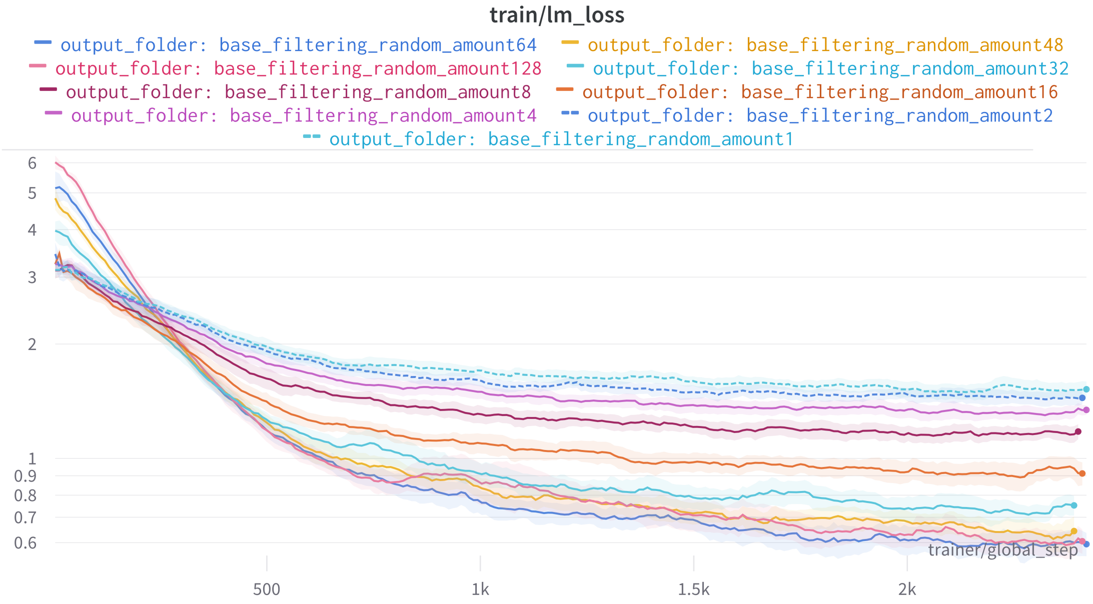


Figure 5-3: Training loss plots for varying amounts of commonsense assertions. We see that typically, the more assertions that are given, the lower the loss will be, indicating that more knowledge in the task tends to make it easier.

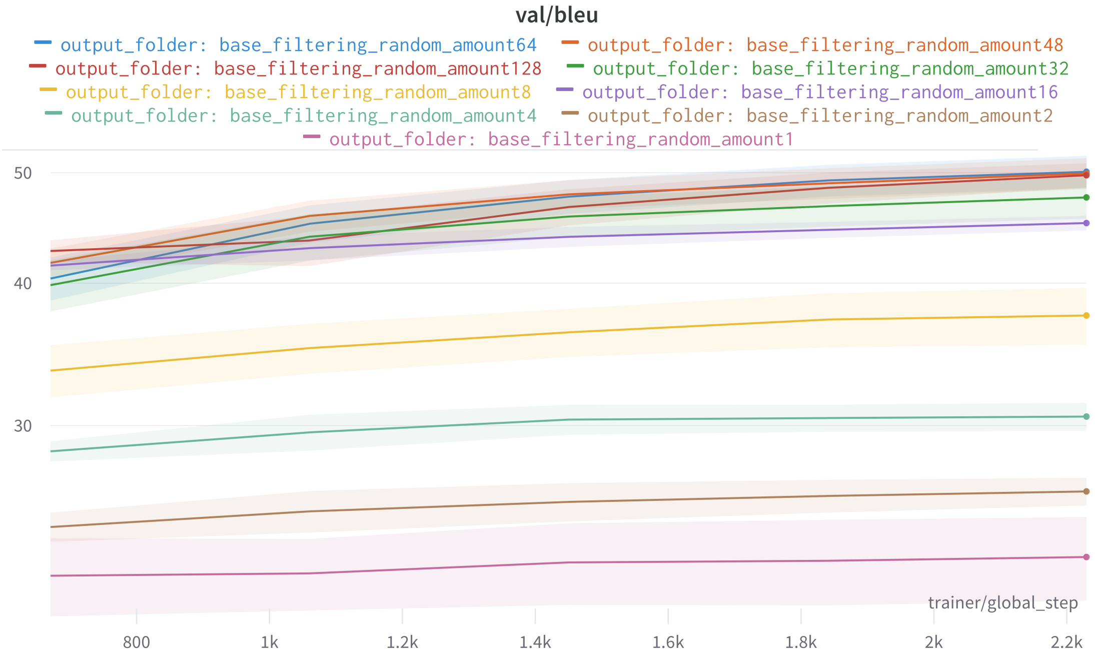


Figure 5-4: Validation BLEU results for varying amounts of information. We see that performance keeps increasing as we increase the amount of information. Interestingly, the performance with 48 facts tends to be at first better than that of 64. We see that at the final epochs, the performance of 64 narrowly beats that of 48. In the tables that we present, this does not seem apparent.

From our results, there are a few things that we can see, firstly, the more knowledge we add into the process, even with random sampling of it, the better the step generation model performs. The second observation that we can see is that above a certain amount (32), the returns begin to diminish.

Interestingly, some of these observations align well with other more recent and parallel work [82]. In the work by Li et al. [82], they find that a few interesting and relevant observations. The first is that larger models are more likely to ignore contexts, which means that smaller models like the ones we utilize will try to leverage context as much as possible, hence performance continually increasing as we increase the amount of facts. They also find that context noise reduces controllability. What this means in our case is that if our context contains too much noise/irrelevant facts the performance of our task, in particular the generation will probably be less, as the facts do not guide the generation of the model. We have evidence of both of these

	Validation							Test						
Assertion Amount	LM Loss	BLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM	LM Loss	BLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM
1	1.541	23.566	53.723	48.388	27.898	47.485	47.516	1.489	24.873	55.767	50.309	29.705	49.313	49.379
2	1.459	27.033	57.535	53.233	32.783	52.127	52.116	1.429	29.142	59.885	54.539	34.360	53.571	53.574
4	1.279	30.846	60.760	57.596	37.683	56.333	56.388	1.281	32.149	62.037	58.571	39.040	57.251	57.271
8	1.079	38.464	67.371	65.564	46.864	63.966	63.892	1.031	39.833	68.097	66.668	48.743	65.384	65.364
16	0.800	47.149	74.281	73.810	58.386	72.146	72.162	0.880	43.005	71.233	70.645	53.591	68.862	68.865
32	0.669	50.248	76.485	76.913	63.348	74.952	74.970	0.667	51.710	76.963	78.010	64.291	76.205	76.216
48	0.621	53.123	78.753	80.428	68.049	78.486	78.514	0.548	56.775	80.098	81.210	69.243	79.409	79.420
64	0.587	52.906	78.329	79.832	66.927	77.430	77.417	0.608	52.473	78.434	79.817	67.110	77.441	77.453
128	0.595	53.804	78.886	80.637	68.383	78.523	78.544	0.576	54.744	79.640	81.013	68.877	79.057	79.017

Table 5.2: Results for test and validation averages for varying amounts of contextual commonsense assertions. We see the trend that generally, the more assertions given, the better the performance. Around 64, we see that performance starts to become unsteady. This is most likely due to context length constraints, since we are using the FlanT5 model, the context is limited to 1024 tokens.

behaviors in the prior tests.

Does the quality of the knowledge matter?

By now, we have seen that we need knowledge for procedural step generation, and we need a certain amount of it to perform well in our task. Now, we explore if the quality of the information matters. To test this, we run three cases: we do not order the information (i.e., we randomly sample facts), we order the facts based on their rouge vs. the procedure step (i.e., which facts have more surface overlap with the step), and we order them based on the plausibility score given by the discriminator (i.e., which facts are more plausible for the given steps).

Once more, we perform our tests on a subset of 1000 procedures that are the most similar to the query “smartphones, software, apps”. We run our experiments with a similar setup: 2 seeds for every one of these test cases, each for 3 epochs. Additionally, we utilized a batch size of 8 and a learning rate of 4e-5 with a linear decay with 20% of the batches as warm up. Additionally, we use a train/dev/test split of 80/10/10. The results of these experiments can be seen in Table 5.3 and in Figure 5-5. We select 64 as the amount of knowledge for this.

From this, we can see that there is a clear benefit to ordering the information. We see that the ordering that is based on how close it is to the ground truth (rouge scoring) tends to be the best performing, while the one based on plausibility is a close second, and random ordering performs inconsistently. From all these tests, we gather that the inclusion of knowledge is necessary, the order of it matters, and the quantity

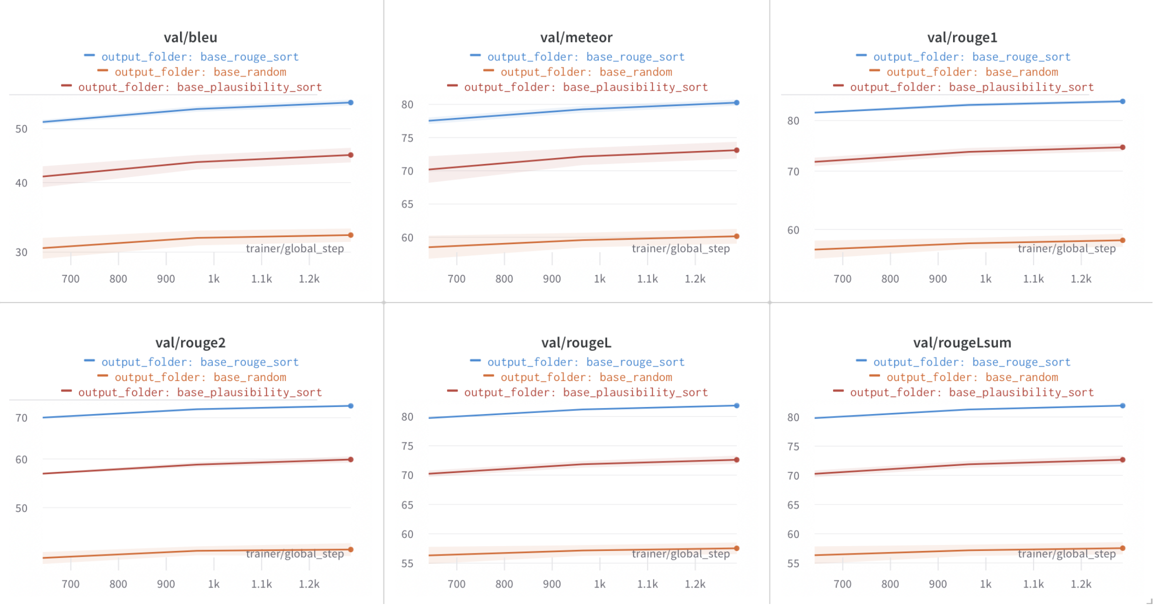


Figure 5-5: Validation metric results for different ordering schemes. We see that the rouge sorting gives the best performance, as it is a metric that can see the output, however the plausibility-based scoring gives the second best ordering.

	Validation						Test					
Name	**BLEU**	**METEOR**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-LSUM**	**BLEU**	**METEOR**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-LSUM**
Random Ordering	32.849	61.155	59.124	43.106	58.187	58.176	35.121	62.625	60.622	45.552	59.703	59.676
Plausibility Sorted	47.128	74.772	76.124	61.862	73.943	74.026	50.555	76.718	78.128	65.275	76.477	76.456
Rouge Sorted	58.311	82.045	85.550	74.806	83.023	83.096	60.178	82.916	86.782	77.235	84.468	84.468

Table 5.3: Results on the validation and testing set for different ordering styles. We see that the rouge-based filtering provides the best performance, but note that to get this score, we need to know the step we are going to generate ahead of time. Whereas random ordering gives the worst performance, and plausibility-based ordering gives an intermediate performance.

of it matters.

Complete training

Before we dive into our explorations of the reasoner, we finally train a procedural step generation model on our full (contextual knowledge graph generated) dataset. For this training, we use a batch size of 2, and we accumulate the gradient for 4 steps, which gives us an effective batch size of 8. We train for 5 epochs with a plan size of 64, and we use the rouge-filtering with random sampling.

We perform our test with a learning rate of 5e-5 with a linear decay with 20% of the batches as warm up. Additionally, we use a train/dev/test split of 80/10/10. The results of these experiments can be seen in Table 5.4 and in Figure 5-6.

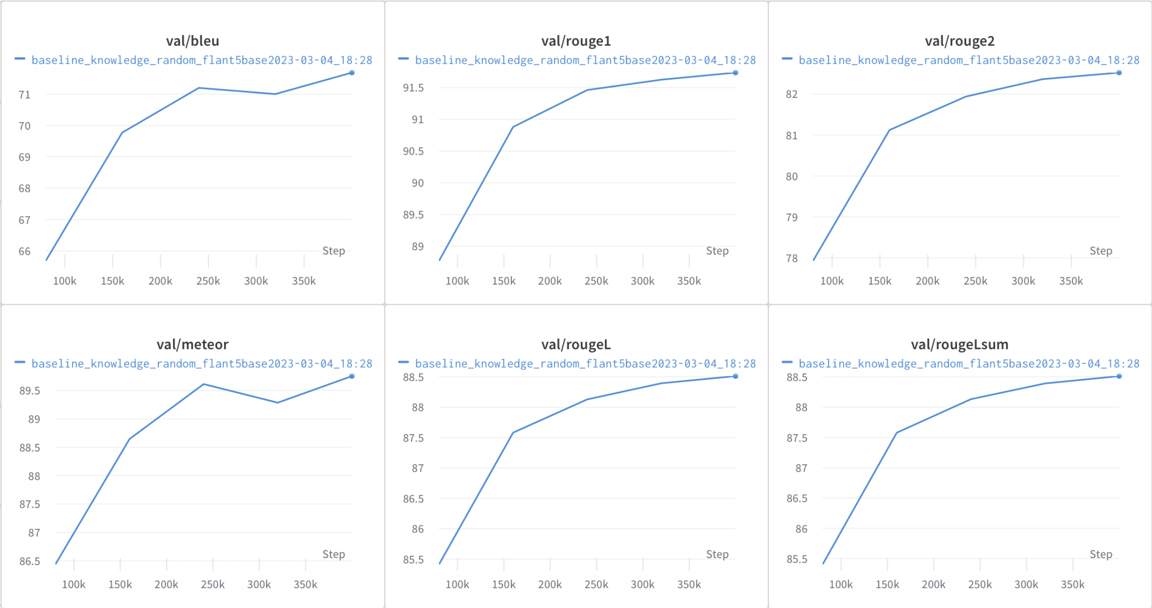


Figure 5-6: Validation results of complete model training. We can see that by supplying useful information and enough procedures, even on the first epoch we achieve good performance on a withheld validation set.

Split	Language Modeling Loss	BLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM
Train	0.042	-	-	-	-	-	-
Val.	0.345	71.686	89.754	91.736	82.524	88.513	88.512
Test	0.338	72.246	89.955	91.881	82.931	88.742	88.749

Table 5.4: Results of complete (with the procedures that have a generated knowledge graph) training, and random sampling of knowledge with rouge filtering.

From these results, we can see that we have a fairly strong model that can indeed generalize to procedures that it has not seen.

5.2.5 Conclusions

From our results, we see evidence of few trends. Firstly, the task of procedural step generation requires additional knowledge, such as the one that is provided by contextual commonsense inference. Secondly, the amount and quality of the knowledge greatly impact the performance of an encoder-decoder model in the task; higher amounts of quality knowledge positively impact the model, whereas noisy knowledge gives some benefits, but they are not as large as utilizing cleaner knowledge. Thirdly, we see that if we keep the amount and quality of knowledge constant, that by varying the order of the knowledge we can impact performance; however it is not immediately clear how to order the knowledge.

5.3 Reasoner

With all the experiments we ran, it was evident that there is a need for ordering/selection of high quality knowledge. We now discuss the approaches that we tried and the results that we achieved with this.

5.3.1 Reasoner Models

Rouge-filter approximation

Since the rouge filtering approach worked so well, we first began by testing out a way of regressing on this rouge score. The reasoner has to, given the procedure title, the step number, and a fact, give an estimate of the overlap between that input and whatever internal representation of what the target step is. For this model, we utilized the mean squared error as our metric to evaluate the system. We also look at a simpler formulation in which the model has to do binary classification of a fact of whether it is in the rouge filtered set or not. We utilized the F1 score for the binary classification as our metric to evaluate the system.

As before, we perform our tests on a subset of 1000 procedures that are the most similar to the query "smartphones, software, apps". In this set of experiments, we run 1 seed for 3 epochs. Additionally, we utilized a batch size of 8 and a learning rate of 4e-5 with a linear decay with 20% of the batches as warm up. Additionally, we use a train/dev/test split of 80/10/10. The results of these experiments can be seen in Table 5.5 and in Figure 5-7 and 5-8. From our results, we can see a few things. Firstly, the simplified task of doing binary prediction is apparently quite a complex task. The model is hardly able to achieve an F1 score higher than 0.7. We would have wanted a score of 0.8 or higher to have some confidence in the classification system.

Task	Validation F1	Validation MSE	Test F1	Test MSE
Regression	-	0.011	-	0.011
Classification	0.671	-	0.677	-

Table 5.5: Here we see the actual values for the metrics used to evaluate whether the approximation for the rouge score for filtering would be effective.

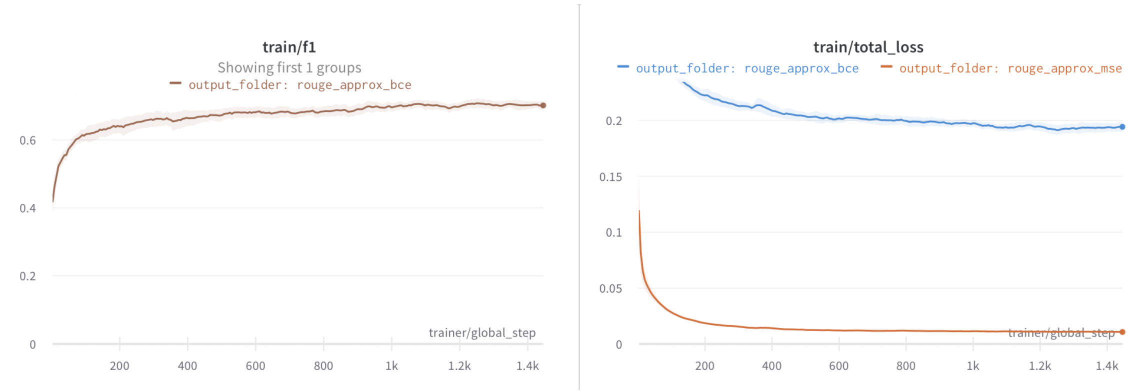


Figure 5-7: Here we see the training metrics for the approximation for the rouge score filtering. We see that the F1 performance plateaus, similar for the MSE loss.

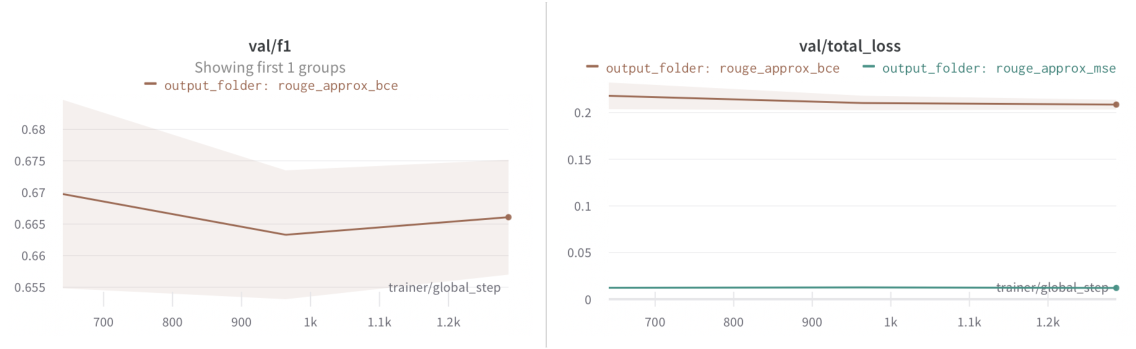


Figure 5-8: Here we see the validation metrics for the approximation for the rouge score filtering. We see that the F1 performance plateaus, similar for the MSE loss.

Similarly, the regression of the rouge score does not go below a mean squared error of 0.01 which roughly translates to an average error of 0.1. From some exploration of finding the average values of the rouge score used for filtering in our experimental dataset, we see that the score averages around 0.2. What all of this together means is that the model is also unable to regress on the score. We believe that the reason for the performance is that the model is not able to generate an effective representation of what the steps in each procedure are. Because of this, there is no effective way to approximate overlaps between the step representations and the input given to the model. Future work should explore this further, as a good approximation for this would give (from our previous tests) good results. Because of these two reasons, we move on to another model for the reasoner. We use mean squared error to calculate the loss for the cosine distance approximation and use binary cross entropy for the confounder loss.

Grounding the Contextual Commonsense Inference

So far, the experiments that we were running contained all the possible assertions that we were generating. What this means is that for a given step, there may be an assertion about a keyword/(noun/verb) phrase that may have been at some other point in the procedure. We decided to tackle this issue and see how we could improve it. The reasoning for trying to improve this was that if we could give the reasoner model cleaner data, it may be able to organize it more effectively than trying to sort out between facts that may be related or useful for a step and those that are not. To address this, we did the following simple change in our contextual commonsense assertion process: we only generated assertions whose hints contained noun/verb phrases that were found in the step. Although this may be a relatively simple thing to do, we found that it decreased the amount of generated facts to be roughly the same amount as when we performed the rouge filtering. We additionally did a small test on our 1000 procedures subset, and found that the overlap between the procedures that were generated with this strategy and those that were left after the rouge filtering was of 80%, indicating that by just constraining the generation, we may not need much more for having a high quality set of data for procedural step generation.

Plausibility: A Strong Baseline Reasoner

After grounding our contextual commonsense inference, we performed a small experiment. We decided to sort our assertions based on the discriminator score from our model. By doing this, we have a simple metric to sort generated assertions and can pick the top j ones. We find that this surprisingly simple strategy, seen originally in our initial experiments of determining that ordering was necessary, serves as an incredibly strong baseline. Evidence of this performance can be seen in Table 5.3, under the "Plausibility Sorted" row. We also perform a small test on 100 procedures to find that if we pick the top 64 plausible assertions, roughly 73% of them (47/64) will be in the top 128 rouge-sorted assertions.

Experimenting with REINFORCE

From surveying the literature of data-to-text, we found the work by Yang et al. [183] to be the most similar to what we would like in our reasoner system. They develop a reasoner that learns to select assertions into a plan $\mathcal{P}$ through reinforcement learning. Their approach is fascinating in that the planner can adapt a plan as the text is being generated. If we were to apply this to our scenario, the planner could adapt to a change in goal when generating steps for a procedure. This would facilitate generating contextual error handling for our work. We do make note that the effects of the planning, at least in the dataset that the authors evaluate on seems to be marginal; they attribute it to a small record vocabulary which makes the model learn the data distribution easily and that the effects of planning are not as evident when evaluated on a sentence-level evaluation (i.e. the planning mechanism is more appropriate for long text generation). What this may mean in our case is that although the planner may provide a boost in performance, if there is a good enough plan such as the plausibility-based sorting, that the improvement may be marginal.

Because of all this, we explore implementing a similar structure in our work. In our work, we implement the record encoder with a sentence-transformer model to

$$
\begin{aligned}
\mathcal{J}^{unsup} = \; & \log P(Y|X,\mathcal{P}) \\
& + \lambda \sum_{k=1}^{K} (r\,(g_k|o_k) - v_1) P(g_k|o_k) \\
& + \lambda \sum_{k=1}^{K} (p(\tilde{g}_k|o_k) - v_2) P(\tilde{g}_k|o_k)
\end{aligned}
$$

Figure 5-9: Objective function for the unsupervised planner in [183]. We see three terms, the first being the language modeling loss of the system, the second being the term that gets maximized which is the likelihood of the plan given the likelihood of the generated text, and the third an exploration term which is if the likelihood of the generated text has not been maximized, there may be an unseen plan that may be explored.

generate a contextual embedding of the record and then encode all records with a bidirectional Gated Recurrent Unit (GRU)[28]. For the planner, we also utilize a GRU, however we only use the static version of the planner, since our output is of 1 sentence. The authors utilize the regular language modeling loss, in combination with a reward function that consists of maximizing the likelihood of the selected plan and scaling it with the likelihood of the generated sequence. They also utilize a reward term that consists of the likelihood of unseen plans to encourage exploration in the system. This objective function can be seen in Figure 5-9.

We tried this approach utilizing a BART model as a realizer (i.e., the graph-to-step model that me mention in a previous section), and we also incorporated the goal context (procedure name and the number of the step) by concatenating them to the fact embedding so that the planner would take them into consideration when selecting. We also looked at the work by Deng et al. [39] on utilizing reinforcement learning to be able to learn an efficient prompt for a language model on a task. Although this may seem like a very separate topic (of prompting vs. data-to-text), we note that the authors are looking for a system that can find a suitable textual input (which in our case can be seen as a plan) for a language model in a language generation task (which in our case is the graph-to-text task). We utilize their key finding that by utilizing the z-score of the reward batch, instead of the direct reward from the reinforcement learning, that it stabilizes the training. We tested this system on a subset of 100 procedures for 5 epochs, with a plan size of 64, and utilizing the realizer as a frozen, pre-trained on random golden knowledge, model that described previously to generate the next steps. What we found is that performance increased inconsistently throughout the epochs. We see this in Figure **??**. We ran this system with 200 procedures and for 15 epochs, and obtained the results found in Figure 5-11.

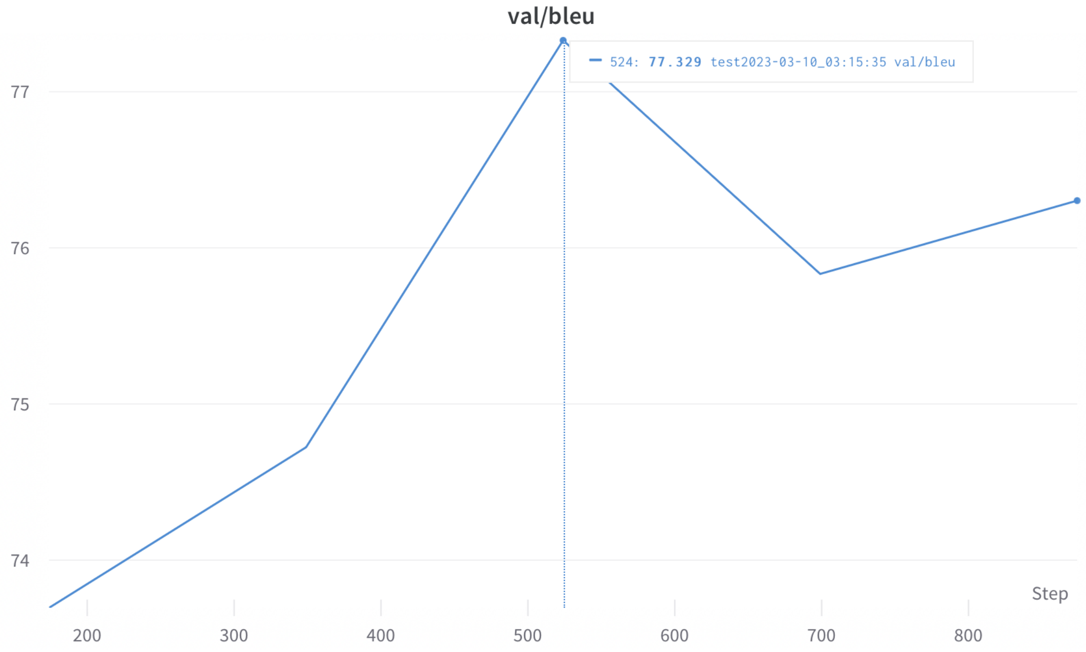


Figure 5-10: Performance of reinforcement learning planner with frozen, pre-trained realizer. The system somewhat inconsistently improves in performance

However, our run with the 15 epochs and 200 procedures took approximately 2 hours, and the one with 100 procedures and 5 epochs took approximately 30 minutes. We note that in our dataset we have approximately 30k procedures, and that reinforcement learning usually takes hundreds if not thousands of epochs to converge. If we were to run this system with all our data for one epoch it would take 600 minutes to run one epoch. If we were to run 15 epochs as before it would take 9000 minutes or approximately 6 days. This is too long for a system that is not consistent in its behavior. The reason that the reinforcement learning process is slow, is that it has to generate a sentence based on the plan that it is giving the step generation model to be

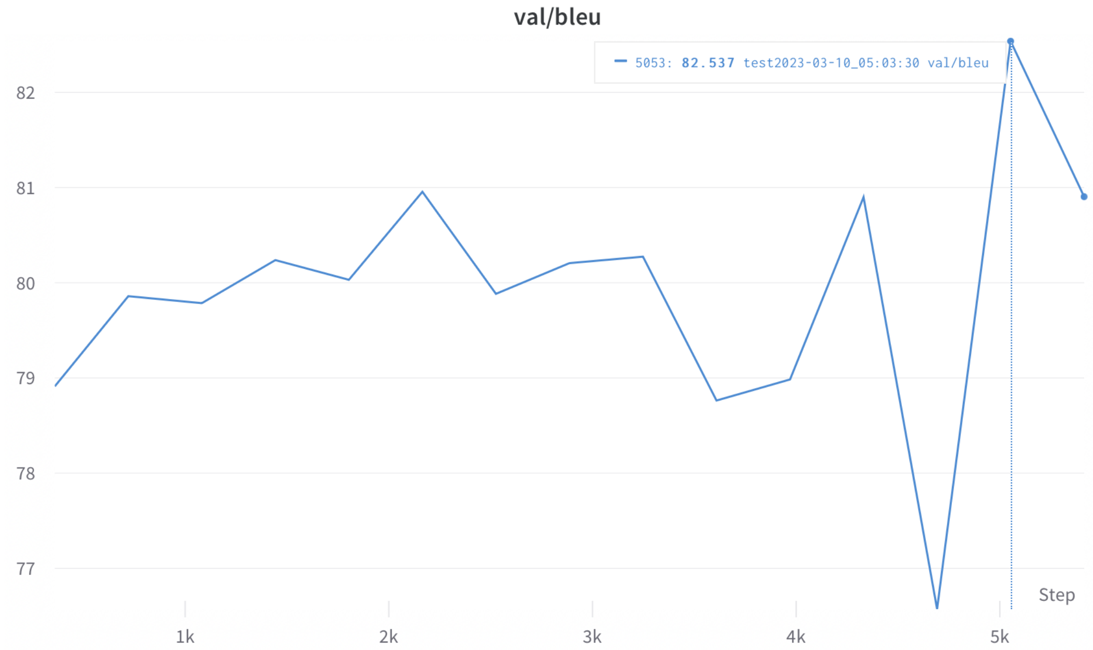


Figure 5-11: Performance of reinforcement learning planner with frozen, pre-trained realizer, for more data and a longer amount of epochs (15 vs. 5). The system somewhat inconsistently improves in performance

able to receive a reward. Although this can be optimized with greedy decoding rather than other alternatives, it might have negative effects on the reward signal that is generated. From all of this, we determine the following. There is some evidence that a reinforcement learning agent can actually learn a plan that is better than sorting by plausibility; however this process is somewhat inconsistent and is too slow to test for hundreds of epochs. We leave it as future work to optimize this into something more useful.

Experimenting with LLM-based planning

While exploring alternatives for a planner, we wanted to see if there was a way of leveraging the contextual power of LLMs for this task. To test this, we devised the following prompt:

> *"Which (minimum [MINIMUM AMOUNT] and maximum [MAXIMUM AMOUNT]) of these facts, as they are written (verbatim), are needed for*

> *step [STEP INDEX] in the procedure "[PROCEDURE NAME]"? Respond with only a numbered list that avoids duplicate facts, if there are any Unicode characters replace them with textual descriptions, avoid facts that state the step verbatim, and only if necessary, add facts that may be missing. [FACT LIST]"*

Here, [MINIMUM/MAXIMUM AMOUNT] represent the minimum and maximum amount of facts that the plan can contain. We set this to 48 and 64 respectively. [STEP INDEX] is the index of the step that we are going to generate, and [PROCEDURE NAME] is the name of the procedure that we are generating the step for. Lastly, [FACT LIST] is a list, separated by new line characters, of the assertions. In our work, we pick the top 256 facts to supply for this list. The reason is that we utilize the ChatGPT API which has a maximum context size of 4096. We find that with more than 256 facts, the chances of exceeding the maximum context length were high, seeing as the system has to repeat 48-64 of these facts.

To test this system out, we pick a small subset (100) of sub-procedures, and sample randomly 40% of steps and check the overlap of the LLM-based reasoner with the top 128 facts sorted by the rouge score. We note that roughly 63% of the assertions selected by the model are in the set of the top 128 rouge-sorted assertions. What this means is that more than half of what the LLM model selects is likely to be useful for the reasoner. Because of this and the aleatory nature of the generation process for the LLM, we use this reasoner as a backup in case that the first generated procedure (based on the plausibility) does not produce a satisfactory result. We note that this is a relatively naive approach. There is no fine-tuning or prompt engineering approach involved in this.

Experimenting with Retrieval Augmented Generation (RAG)

We explore one last alternative for both planning and generation. Retrieval based generation methods have recently gained popularity. They serve as a simple way to augment a generative language model with information that it may not have seen during its pre-training, or with updated information [129, 57, 80]. We opted to test out the RAG [80] model to see how it performed in our task. We utilize the RAG-token model, because it permits conditioning the text generation on multiple source documents. Now, if we take the source documents to be facts, and the text generation to be the task of procedural step generation, we see that it is possible to adapt the RAG model for our work. We generate a small dataset, consisting of 100 procedures along with the steps and their contextual commonsense assertions, without the grounding. We treat the assertions as individual documents for the generation task. Although this might be a slightly more complicated scenario than having the relevant step subset of assertions, it is a more interesting scenario in that it has to look through all the facts that it has to find relevant ones for the procedure. This is a desirable behavior, because it would permit completely dynamic procedure step generation. With a single title for a procedure, we could generate all of its steps without having to know which knowledge belongs to it. We find that the RAG model in this manner underperforms. We train it for 5 epochs and find that it is able to reach a maximum validation BLEU score of 27.437. We note that the validation loss in these runs kept increasing and that the BLEU score kept decreasing as epochs went on. We note that for future work, more exploration of the RAG model should be done, as it would permit much more flexibility come inference time.

5.3.2 Conclusions

After exploring a variety of ways on how to reason with the contextual commonsense assertions (i.e., contextual procedure knowledge graph) we determined that the simplest approach is to ground the generation process on noun/verb phrases, and to sort these grounded assertions by the plausibility score given by the contextual commonsense inference discriminator. With this strategy, we now have a way to be able to convert a knowledge graph into a procedure. We do note however that there is an assumption that the graph can be split into the portions that correspond to each step. At the moment, there is no other work that has explored this idea of graph-based procedural step generation, so we take this for granted. We note that in the future, a more robust method that only requires the goal/name of a procedure, would be able to automatically look up, from all our generated assertions, the ones that would be useful for generating the steps for a procedure. It would essentially perform reasoning on a scale that would be considerably more ambitious.

5.4 Procedure Error Handling

Up until now, we have described the process to be able to generate the steps for a certain procedure from a knowledge graph. We now address what the process is to generate the steps for an exception or error within a procedure.

5.4.1 Algorithm

Broadly, the algorithm for handling errors within procedures can be seen as an extended telling within an extended telling. An extended telling is a conversational pattern in which an agent needs to convey to a user a large amount of information, and does so by splitting the information into bite sized chunks and relaying them to a user. The complete pattern and an example can be seen in Section 6.2. Extended telling can be utilized for telling stories and giving instructional guidance on a procedure. In the case of a procedure, the information is split into steps and conveyed to a user. However, it may be the case that a user performs a step incorrectly as the guidance is being given. In many cases, a simple clarifying question (which can be seen as a repair operation) can be asked to a conversational agent, and guidance can continue. However, we think about the case where such a question is not enough, and a person would need to embark on a sub-procedure to fix the mistake. Visually, this can be seen in Figure 5-12.

We treat a request for help within a certain procedure as a recursive process. However, we have some additional information which is the context in which we were in: the step where the error occurred, and the procedure that we were attempting originally. With this, we can make a query for the procedures that we know in a way that is structured as: “How to fix [ERROR] when doing [ORIGINAL PROCEDURE]”. After this, we can follow the same process of extracting steps and generating the graph

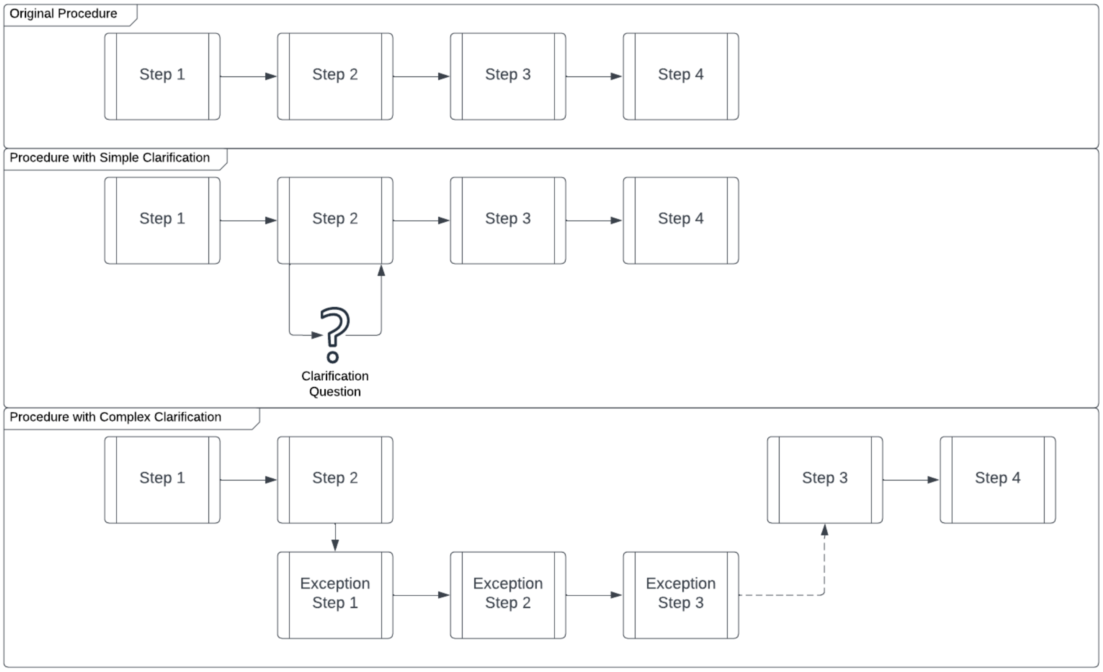


Figure 5-12: Here we can visually see how a sub-procedure exception can be handled. It can be handled either with a simple clarification question which lets a user proceed to the next step, or with a sub-procedure to fix the error. In the case of the sub-procedure, it may also be possible to complete the process in an attempt to fix an error. This can be seen as an alternative way of doing the procedure, and is the reason why there is a dashed line (possible discontinuity) in the end of the exception process.

as in Section 4.7.3 and 4.7.4.

We note that we can additionally utilize the prior context when generating the instructions. To do this, we interleave the knowledge from the last step that was done, with the knowledge of the first t steps in the procedure. To give more importance to using prior knowledge at the beginning of the exception procedure, we exponentially decay the amount of information that is interleaved as the exception procedure continues until the ith step of the exception procedure, where we stop interleaving knowledge. We note that we can do a similar process to incorporate the knowledge of the original upcoming step, to gradually add more facts in until the last step of the exception; however, we do not implement this in our work. By interleaving the knowledge from the original procedure, we empirically see that some aspects of the exception instruction take into consideration the context that was being done before. One example is on the writing a check procedure. If a person puts the wrong date, then the person needs to correct the date mistake on the check. If we look at a procedure for correcting a check, typically the mistake is crossed out and initialed. If we generate an exception step without the context that a person was writing the date, the knowledge based generation would look like this:

Step 1: Cross out the mistake and write the correction on the check.
Step 2: Write your initials next to the corrected mistake.
Step 3: Write your check again if it hasn't been accepted or corrected.

Although it is a perfectly acceptable error handling procedure, the steps are not entirely contextual to it being a mistake about the wrong date. If we included some facts, especially ones that indicate that the procedure was related to the date, and we interleaved them, with the ones on how to fix a mistake, the generated step would look like the following:

Step 1: Cross out the date and write the correction.
Step 2: Write your initials next to the corrected date.
Step 3: Write your check again if it hasn't been accepted or corrected.

We can see that the mistake is contextualized to be the date, and would make it more straightforward for the user. We utilize this strategy in our error handling. Future work should explore improved contextualization of errors.

5.5 Summary

In this fifth chapter, from our experimentation and our results, we can say that *it is* possible to generate steps for procedural guidance by formulating the problem as a graph to text task, in which the graph is generated from a contextual commonsense inference model and the text is the procedural step. We also find that the quality and amount of information greatly impacts the generation system. Because of this observation, it is highly recommended that the system is split into a planning stage and a generation stage, in which the planner tries to come up with a useful plan for the generation. For the planner in this task, we explored unsupervised, semi-supervised, and reinforcement learning systems and found that they all unperformed when compared to the simple baseline of ranking the knowledge based on the plausibility of the context it was acquired from. We also explored utilizing a LLM as a proxy for a reasoner, and found that it is an interesting possibility, however it has a limited context that it can process. Following this, we explore and devise a simple recursive algorithm to be able to handle exceptions / issues within procedures, and explore the effects of combining knowledge from different procedures to contextualize the error handling.

Chapter 6

Relaying Steps through Conversational Agents

The first ingredient in conversation is truth, the next good sense, the third good humor, and the fourth wit. -William Temple

6.1 Overview

Up until now, we have described a system that is capable of generating procedural steps from a knowledge graph about a procedure. If we simply gave these instructions as text, we might run into the issue that people find that reading instructions can be intellectually taxing. People often prefer to be guided through a procedure by another person [44]. To address this, we decided to give these instructions to a conversational agent. Conversational agents provide a natural and intuitive way for people to receive guidance on completing tasks. Conversational agents simulate human-to-human interactions and present information in a more natural and intuitive way. They allow people to interact through speech, which is widely available and less taxing than reading[88, 9]. In this chapter, we describe how a conversational agent would convey instructions generated instructions through patterns from the natural conversation framework [108]. We then describe how to build two conversational agents, one utilizing intent-based actions, natural conversation patterns, and our generated steps, and one that is zero-shot based on large language models (ChatGPT).

6.2 Conversational Patterns

Although it may seem trivial to just give users the steps, there are certain principles and structures that are needed for conversations to both flow and feel natural [108]. Through Conversation Analysis (CA), certain patterns of interactions have been observed that help to inform the design of conversational agents. The Natural Conversation Framework (NCF), developed at IBM Research [108] is a systematic framework for designing interfaces that work like natural conversation. It consists of four main components: 1) an interaction model of "expandable sequences," 2) a corresponding content format, 3) a pattern language with 100 generic patterns and 4) a navigation method of six basic user actions [108].

Altogether, patterns from this framework, along with guidelines of how content should be structured, can be mixed and matched to construct a conversational agent for a task such as procedural guidance. We now give an overview of key conversational patterns and how they could be applicable to our procedural system. Next to the name of the pattern, we give the identifier for it found in [108] in case the reader wants more details. The combination of these patterns would serve as a foundation for a general conversation regarding a procedure and some of the possible events that occur within it.

Extended Telling – A3.0 The base pattern that we will utilize for our conversation system is the *Extended Telling* pattern. This pattern describes how a conversational system can break down a large body of information (e.g. storytelling and instruction giving) into manageable chunks that a person can work with. From [108]: "[The interaction is usually started by phrases such as] "how do I meditate?," "how do I do that?," etc. The telling is then done as: the teller gives the first part of the telling and then waits for an indication from the recipient to continue the telling. Such "continuers" [156] may be verbal tokens, like "uh-huh," "mhmm," "okay," "all right,"

Turn Number	**Turn Agent**	**Action**	**Example**
1	User	STORY REQUEST/INSTRUCTION INVITATION	How do I meditate?
2	Agent	PART/STEP 1	First, sit comfortably and breathe slowly.
3	User	CONTINUER/PAUSE	Ok
4	Agent	PART/STEP 2	Next, count each in breath and out breath until you get to ten. Then repeat.
5	User	REPAIR INITIATOR	What do you mean?
6	Agent	REPAIR	As you breathe in, out, in, count one, two, three all the way to ten.
7	User	CONTINUER/PAUSE	All right
8	Agent	PART/STEP 3	Finally, continue counting your breaths for ten minutes.
9	User	SEQUENCE CLOSER	Thanks
10	Agent	RECEIPT	You're welcome!

Table 6.1: Example of extended telling pattern (A3.0) taken from [108]

“yeah,". In our work we utilize, in addition to this, variations of “Yes”, “Done”, “Go on”, “What’s next”, “Continue”,“keep going”, “next step”, among others as continuers. An overview along with an example how-to of this pattern with repairs [108] can be seen in table 6.1. This is the main pattern that handles our procedural interactions. It describes broadly how the interactions occur, and it describes what happens when there is a misunderstanding in a step. However, it does not describe the case that there is an error in a procedure. The way we treat that we describe later as the error handling procedure. Briefly, errors occur in the form of an inquiry (explicit as “I made a mistake, what do I do” or implicit “I made a mistake (How do I fix it)”). If the error handling procedure is simple (approximately 2-3 steps or less) we can respond with how to fix the error. However, if it is longer than that, we may enter into a recursive extended telling to relay the instructions of how to complete a procedure.

Repair (Agent) B1.0 The *extended telling* pattern requires that an agent is capable of both initiating and handling repair procedures. This conversational pattern and following one describe how an agent can initiate a repair to clarify steps for a user (among other repair interactions). In the Conversation Analysis literature, “repair"

Turn Number	**Turn Agent**	**Action**	**Example**
1	User	<UTTERANCE>	whats shakin bruh
2	Agent	<PARAPHRASE>	You mean, what's new?
3	User	CONFIRMATION	yeah
4	Agent	<RESPONSE>	Nothing really. I don't get out much.
5	User	SEQUENCE CLOSER	ha!

Table 6.2: Hearing Check (B1.1.0) pattern taken from [108]

is a technical term referring specifically to the repeating and paraphrasing of prior utterances or parts of them. [108] In this specific pattern these requests originate from the agent, in the following pattern, they initiate from the user. Within this larger pattern, there are two different repair mechanisms: hearing checks (B1.1.0) and paraphrase requests(B1.2.0). These can be seen in tables 6.2, 6.3,respectively. We note that this strategy can be employed also whenever a user asks the system a procedure that it may not know exactly, in the case of a simple curated procedural lookup system.

Turn Number	**Turn Agent**	**Action**	**Example**
1	Agent	<ANY UTTERANCE>	Hello. How are you?
2	User	<UNKNOWN UTTERANCE>	bitchin'!
3	Agent	INITIAL PARAPHRASE REQUEST	What do you mean?
4	User	<KNOWN PARAPHRASE>	I'm great!
5	Agent	<APPROPRIATE RESPONSE>	That's great!

Table 6.3: Paraphrase Request (Agent) (B1.1.0) pattern taken from [108]

Repair (User) B2.0 The previous pattern describes how an agent can initiate a repair to achieve understanding. However, it is more than likely that a user will also initiate a repair. Users may not understand what the agent is saying, and may require paraphrases, or they may not have heard what the agent said and may require a repeat. Handling repairs is a critical aspect of conversational competence. [108] Here we focus on repetition and paraphrasing requests (B2.1.0 and B2.4.0), which on a preliminary user study that we will describe in a coming section seemed to be the most useful/requested.

Turn Number	Turn Agent	Action	Example
1	User	<ANY UTTERANCE>	I guess I like movies with a strong AI lead.
2	Agent	REPEAT REQUEST	what did you say?
3	User	REPEAT	I guess I like movies with a strong AI lead.

Table 6.4: Repeat (User) B2.1.0 pattern taken from [108]

Turn Number	Turn Agent	Action	Example
1	Agent	<ANY UTTERANCE>	I guess I like movies with a strong AI lead.
2	User	PARAPHRASE REQUEST	what do you mean?
3	Agent	PARAPHRASE	I enjoy movies in which the main character is an Artificial Intelligence

Table 6.5: Paraphrase Request (User) B2.4.0 pattern taken from [108]

Although simple repetitions may be effective at clearing up hearing troubles, the user may not be satisfied or may not have reached understanding with a simple repeat of what was said. A paraphrase may be needed to restate the agent's utterance in a way that a user could understand. Observing the principle of minimization, the system's initial utterance should be concise, but if a particular user indicates a trouble in understanding, minimization should be relaxed and a more understandable version should be offered.

In our use case, repairs in the conversation while in an extended telling, may not be as simple as a question/answer pair; a repair may lead to sub-procedures, in which case they would once more be handled by the *extended telling* pattern.

Inquiry (User) – A1.0 This pattern gives a description of the basic conversational sequence that would occur with a conversational system. It describes what happens when a user asks an agent an inquiry. In our procedural case, we can leverage the intermediate knowledge graph to attempt to give an answer, however we simply defer to a LLM to handle this, and leave as future work leveraging the intermediate graph. We opted to do this, because the inquiry may be more than broad than just about a procedural step.

Turn Number	**Turn Agent**	**Action**	**Example**
1	User	INQUIRY	what computer won Jeopardy?
2	Agent	ANSWER	An IBM computer named Watson.
3	User	SEQUENCE CLOSER	ok.

Table 6.6: Inquiry (User) A1.0 pattern taken from [108]

Turn Number	**Turn Agent**	**Action**	**Example**
1	Agent	OPEN INQUIRY	What did you think of Marios's?
2	User	ANSWER	the food was really good, and I loved the cantina atmosphere!
3	Agent	HEARING CHECK	You said: the food was really good, and I loved the cantina atmosphere! Is that correct?
4	User	CONFIRMATION	yep
5	Agent	SEQUENCE CLOSER	Thank You!
6	User	RECEIPT	you're welcome

Table 6.7: Confirmation A5.2 pattern taken from [108]

Confirmation A5.2 This pattern builds on the Inquiry (Agent) pattern by describing how an agent would react in cases where the information that the agent heard may need to be confirmed before proceeding. In our case, one possible use case would be confirming before embarking on processing a procedure.

Disconfirmation A5.3 Similarly, this pattern would handle when the previous pattern (Confirmation A5.2) was not successful and may involve some repairs (e.g. Repeat B2.1.0) to achieve understanding.

Completion Check - A5.4 We may also need the agent to essentially go through confirming acknowledgement of certain parts and possibly request additional information. This can be done to get different parameters that may be needed.

Open Request A2.0 This pattern is the canonical request from a user. The user requests something and the agent grants that request. In our case, the grant could be handled by the *extended telling* if a user asked for a how-to.

Turn Number	**Turn Agent**	**Action**	**Example**
1	Agent	OPEN INQUIRY	What did you think of Marios's?
2	User	ANSWER	the food was really good, and I loved the cantina atmosphere!
3	Agent	HEARING CHECK	You said: the food was really good and I loved the can I atmosphere. Is that correct?
4	User	DISCONFIRMATION	no
5	Agent	REPEAT REQUEST	What did you say?
6	User	ANSWER	food was good and I loved the CANTINA atmosphere
7	Agent	HEARING CHECK	You said: food was good and I loved the cantina atmosphere. Is that correct?
8	User	CONFIRMATION	yes
9	Agent	SEQUENCE CLOSER	Thank You
10	User	RECEIPT	You're welcome

Table 6.8: Disconfirmation A5.3 pattern taken from [108].

Turn Number	**Turn Agent**	**Action**	**Example**
1	Agent	OPEN INQUIRY	What did you think of Marios's?
2	User	ANSWER	the food was really good!
3	Agent	COMPLETION CHECK	Anything else?
4	User	NEXT ANSWER PART	and I loved the cantina atmosphere
5	Agent	COMPLETION CHECK	Anything else?
6	User	DISCONFIRMATION	no
7	Agent	REPEAT. RECORDING REQUEST	You said: the food was really good and I loved the cantina atmosphere Shall I record your review?
8	User	CONFIRMATION	yes
9	Agent	SEQUENCE CLOSER	Thank You
10	User	RECEIPT	You're welcome

Table 6.9: Completion Check A5.4 pattern taken from [108]

Turn Number	**Turn Agent**	**Action**	**Example**
1	User	FULL REQUEST	can you recommend a nearby Mexican restaurant?
2	Agent	GRANT	Mario's is located at Beach and Main.
3	User	SEQUENCE CLOSER	thanks
4	Agent	RECEIPT	You're welcome!

Table 6.10: Open Request A2.0 pattern taken from [108]

Turn Number	**Turn Agent**	**Action**	**Example**
1	Agent	GREETING. SELF-IDENTIFICATION. OFFER OF HELP.	Hi. My name is Winston. How can I help you today?
2	User	<RESPONSE>	Can you show me how to write a check?

Table 6.11: Opening Self-Identification C1.1 pattern, taken from [108], and applied to our conversational agent

Opening Self-Identification C1.1 This pattern handles the initial conversation from the voice assistant. It presents itself and offers help at the end, leaving the user to respond to the offer for help.

6.3 Conversational Interaction Considerations

For a conversational system to be usable, conversational, and to a certain point feel natural, the following criteria should be addressed in its construction. These criteria/guidelines were taken from [29, 144, 4, 108, 3, 1, 2, 125, 122].

6.3.1 Simple, user-oriented language

If possible, no technical jargon should be used. This means that commands and communication should be with common/naturally occurring words and not uncommon words or phrases. In turn, the system that generates our steps should take its output into consideration and possibly supply synonyms for more complex words, or some explanations if no simpler words appear. However, technical jargon may be introduced to help teach users concepts that may be within certain procedures. In such a case, the term can be used, but repairing of the conversation needs to be available if the user does not understand it.

6.3.2 User control

Users should be able to undo, redo, go back, and even exit without any complicated dialogues. If a user gets into a feature or space they didn't intend, they should be able to leave easily. The system that manages the conversation should be able to exit a sub-procedure or an interaction itself easily if asked by a user.

6.3.3 Consistency in platform

Commands should be consistent throughout the whole application and not context dependent. The system should follow platform standards for Voice User Interfaces (VUIs) if present (e.g. standards for Amazon's Alexa or Google's Assistant). This means that if we have a command (e.g. next step, previous step), this command should be the same in every procedure that we want to guide a user in and not change even if the domain changes.

6.3.4 Error prevention and handling

The conversational system should avoid confusing situations or situations that may be prone to errors. The system should also have methods for repeating, paraphrasing or rephrasing all or parts of a prior turn to clear up any doubts that may be due to hearing misunderstandings, or for explanations that may not be effective. The system should have methods for dealing with errors, misunderstandings, and interruptions in conversation (the system should have some repair sequences which we will see in Section 6.2).

6.3.5 Recognition rather than recall

The system should at some point make actions and options visible or easily retrievable. This would mean that a help function should be present within the application, and possibly “missable" actions should be noted in some way possibly with cues. This way, it would be clear for a user what he/she can do with the system and not have to guess possible actions.

6.3.6 Minimalistic design or principle of minimization

Dialogues should be to the point and try to convey the message in the most effective way possible. This means that the interactions should minimize the amount of words to convey a message. In order to exploit the efficiency built into natural conversation, a conversational agent's responses should be as short as possible while still enabling most users to understand. A simple metric is that turns should be at most a sentence, and in exceptional cases possibly 2 or more.

6.3.7 Error messages

If there are error messages, the user should be able to understand them and try to help the system to snap out of the error states. These messages should not be cryptic or not understandable. If possible, the error messages should help the user to correct course and fix the error.

6.3.8 Help and documentation

The system should be, to a certain extent, usable without any instructions, but a help functionality should be put in place anyways. This help functionality should be easily accessible and relevant to what is being done.

6.3.9 Conversation first rather than visual, system, or content first

The system should be conversation first. In broad terms it means that it should rely only on mechanisms that facilitate communication only through conversation, this excludes relying on screens and other means. The system should keep track of some context of what has been said. This means that additional information that may not be used immediately should be stored appropriately. If the messages the system conveys are not understood some paraphrasing mechanism should be used to try to reconvey them. At a minimum, the system should have the ability to repeat a message. The system should be able to handle some preliminaries (e.g. checking if a capability is possible, introductions, these will be presented in section 6.2). The system should use mostly or purely voice input/output. All of this means that the procedures that we want to be able to help a user with should be doable purely by voice as a minimum, and could be enhanced with visuals if needed.

Caveat Since we are utilizing these conversational patterns for procedural guidance, this requirement can be somewhat flexible. We say this because it may be the case that some instructed learning step is too complicated to understand via voice. An example of this is tasks that include very visual descriptions (e.g., assembly tasks that require a specific piece that is not labeled, repair tasks that require identification of a piece that is not labeled). In these cases, we recommend that some visual is given, whether as a link to the visual or as an image in the conversation app, this may help greatly to achieve completion.

6.3.10 Content formatting

The system should limit its utterances to a single sentence or less whenever possible. It should also break paragraphs down as adequately as possible. It should also let users control the level of detail (i.e. gives more information when asked).

6.3.11 Basic conversational functionality

The system should be able to respond to capability checks (i.e. what can you do?), repeat requests (due to the transient nature of voice), and paraphrase requests The system's utterances should be short and its paraphrases longer. It should also have sequence closers (e.g. thanks) and sequence aborts. It should respond to simple hello and goodbyes. The system should permit non-linear interactions (Expert users should be able to move around with speed-ups/shortcuts) if and whenever possible. Additionally, if possible, pauses when relaying content, fine-tuning the text-to-speech should be taken into consideration.

6.3.12 Consistent persona

The system's persona should be consistent through the application and should be polite to whatever extent possible.

6.3.13 Feedback & Responsiveness

The system should demonstrate that it is working on a task if it takes time or give some kind of immediate feedback at every step. The system's status should also be evident (people should be able to ask what is happening, and the system should answer). If we put this in the procedural interaction domain, the system should be able to give the user information of where they are in the procedure and possibly how many steps are remaining, or in the case that some extensive processing needs to be done, that it notifies the user that it is working on the task.

6.4 Conversational System for Procedural Guidance

We have described subsystems that would take an input query for how to perform a procedure such as "How to write a check?" and look up a relevant page using a web search. Then, utilizing our contextual commonsense inference engine, we generate a knowledge graph for the given procedure, as described in Chapter 4. We then utilize a reasoning system, as seen in Chapter 5 to order the most essential knowledge of this procedure in order to generate a set of steps for the given knowledge. We also demonstrated in Section 5.4, that this same mechanism can be utilized to generate steps for errors by combining the knowledge from a sub-procedure, and contextualizing it by interleaving it and reasoning on it.

With these subsystems, we have the technical means to handle how-to procedures along with exceptions[1]. As a side-note, if we combine the intermediate graph that is generated along with the reasoner ordered knowledge, we also have an interpretable representation of the procedure that we are trying to solve. However, we have not yet described how the system would actually respond to a user nor how a user would interact with the system. We address that now, and note that we will refer to the conversational system interchangeably with the conversational agent.

---

[1] We note that future work may look into an additional styling system may be required to massage the output of the text generation system into something that is more conversational.

6.4.1 Approach Overview

To convey these steps in an understandable and actionable fashion by an end-user we implement a conversational interface that incorporates conversational design patterns and guidelines [108] to have users navigate the procedure, ask questions, ask for repetitions or rephrasing, and for handling exceptions such as not being able to complete a step. Such an interface would make the technology be usable by people, and would hopefully feel like a knowledgeable friend was guiding you through a procedure. We combine the patterns that we mentioned in Sub-section 6.2, the considerations that we had mentioned, along with observations that we have acquired from a small usability user study in the Appendix B.2. into an intent recognition system utilizing the IBM Watson Assistant platform[2] to build a conversational agent to handle procedural interactions.

Within this system, we dedicate intents that attempt to catch when there is an error in a procedure (e.g., a person writes their name erroneously in the amount field of a check). The way this works in a very high level is that within a conversation, a user has to tell the system that something went wrong. The system will then try to parse what went wrong and proceed to find steps for how to fix the issue within the context of the procedure. In the case that the agent does not identify the error, it will try to ask a user for a description of the error. Within the conversational framework, this can be seen as a modification within the extended telling pattern as seen in Table 6.12

We developed the system incrementally, starting with a reference implementation of conversational patterns, and continuing to combine the reference implementation with the knowledge driven generated steps and with open domain question answering. Following this, we incorporated the exception handling mechanism as additional intents and modifications within the patterns. We then tested this system as we describe in the following Chapter, reading out loud a gold set of instructions against our knowledge driven instructions. We additionally tested this approach out against a conversational agent using the ChatGPT model in a zero-shot manner with some

---

[2] https://www.ibm.com/products/watson-assistant

Turn Number	**Turn Agent**	**Action**	**Example**
1	User	STORY REQUEST/INSTRUCTION INVITATION	How do I write a check?
2	Agent	PART/STEP 1	First, write the name of the recipient.
3	User	CONTINUER/PAUSE	Ok
4	Agent	PART/STEP 2	Next, write the date in the upper-right hand corner.
5	User	ERROR REPORT & EXTENDED TELLING REQUEST	I made a mistake, I wrote the wrong date. What do I do?
6	Agent	EXTENDED TELLING (INQUIRY STEP 1) .	No worries! We can fix this. Start by crossing out the wrong date.
7	User	CONTINUER	I did that what now?
8	Agent	EXTENDED TELLING (INQUIRY STEP 2) .	Write the correct date with your initials next to it.
9	User	CONTINUER	Ok
10	Agent	COMPLETION CHECK	Thats it! Should we continue where we left off writing a check?
11	User	CONFIRMATION	Yeah
12	Agent	COMPLETION CHECK	OK then! Write the name of the recipient.
13	User	REPAIR INITIATOR	What's that?
14	Agent	REPAIR	The recipient is the person receiving your check.
15	User	SEQUENCE ABORT	Let's stop for now.
16	Agent	RECEIPT	Ok! If you'd like to continue later just let me know!

Table 6.12: Modified extended telling pattern. This occurs in the case that a simple inquiry triggers a longer explanation of how to do a process.

setup guidance prompts. We now describe the architecture for both of these systems.

6.4.2 Architecture

The architecture that we utilize to power our agent can be summarized as three large components. The first one is a graphical user interface which is where the user either dictates or writes their query, and receives a voice and text response for the agent. The second large component is a backend web-sever that converts a user's conversation into usable input for a selected model (Baseline Gold Instruction Model, Knowledge Driven Procedural Step Generation Model, or a fully automated GPT-3 based agent). The backend is also in charge of querying the third large component: a data-store. The data-store holds the procedural dataset scraped from Wikihow, any procedure that has been looked up from the web-search, and all the knowledge that has been extracted from the contextual commonsense inference model for a specific procedure. This is the ElasticSearch instance that we described in Chapter 4. The need for this is that both the knowledge generation and step generation operation can take time that will slow down a user interaction and possibly frustrate a user, so we try to cache as much as we can.

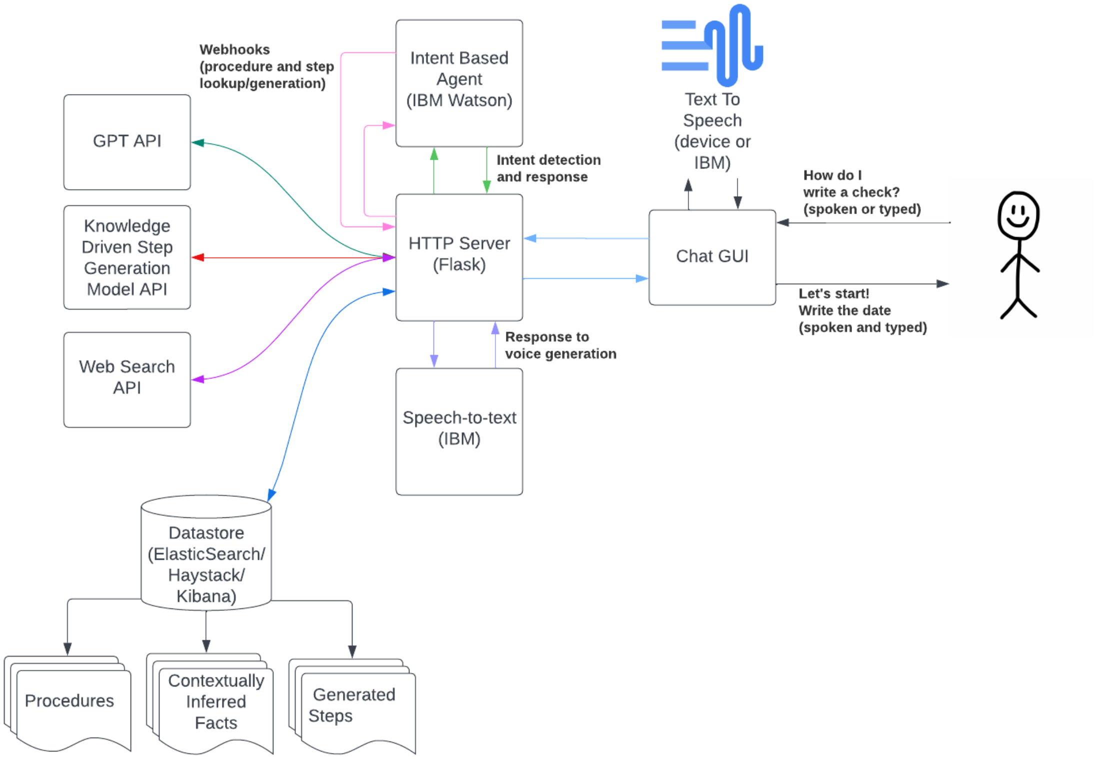


Figure 6-1: Architecture used to power conversational agent for procedures. Users interface with a Graphical User Interface (GUI) which lets them talk or type to an agent. The interface pings the backend server for a selected model (Baseline-Gold Instruction Reading Model, KDPSG-Knowledge Driven Procedural Step Generation Model, or GPT- GPT-3 fully driven agent) and gets a response which it then converts to audio. The user receives the textual response and the audio one (if audio is enabled)

.

Conversational Agent Graphical User Interface

We developed a Graphical User Interface (GUI) utilizing primarily the React-Native, Expo, and react-native-gifted-chat packages. Screenshots of the application are shown below.

The GUI contains 2 primary screens. In Figure 6.4.2, we can see the first of these, which is a login screen. Here, users can log in with an account, or register for one and then log in. The second screen, shown in Figure 6-3 is the chat screen. Here, users can interact with the agent either by typing or dictating by pressing the microphone button. When users log in, the GUI pings the backend to begin the

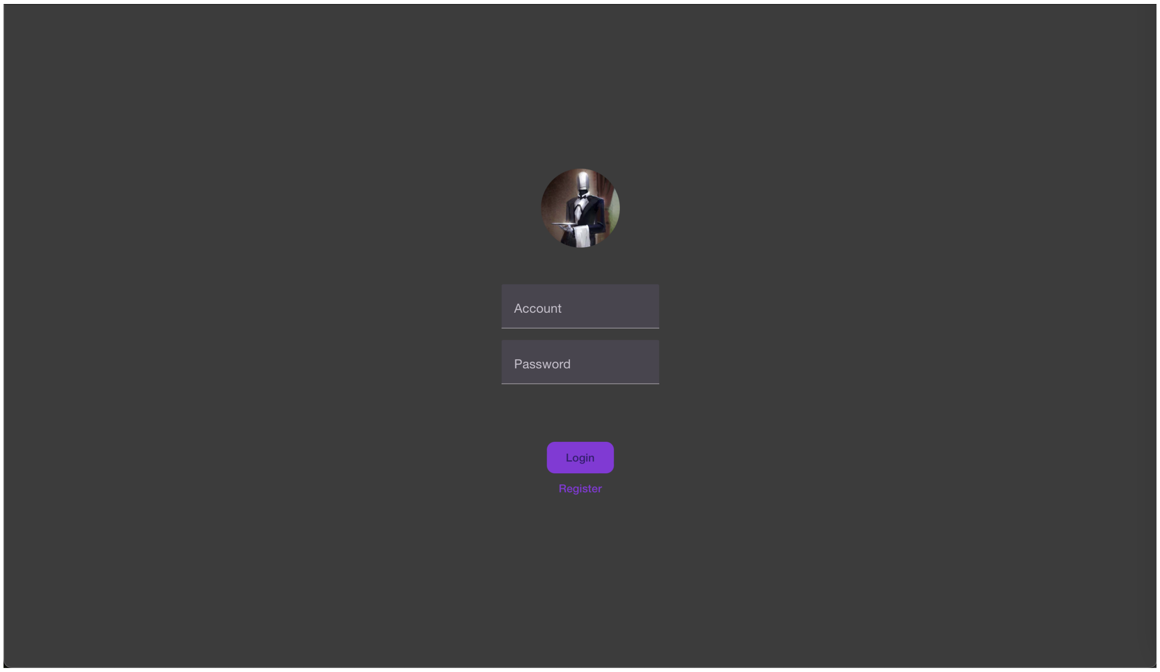


Figure 6-2: The GUI consists of two main screens, the login screen (pictured here) and the conversation screen (pictured below). In the login screen, users can register to have access to a voice assistant or can log in with a registered account

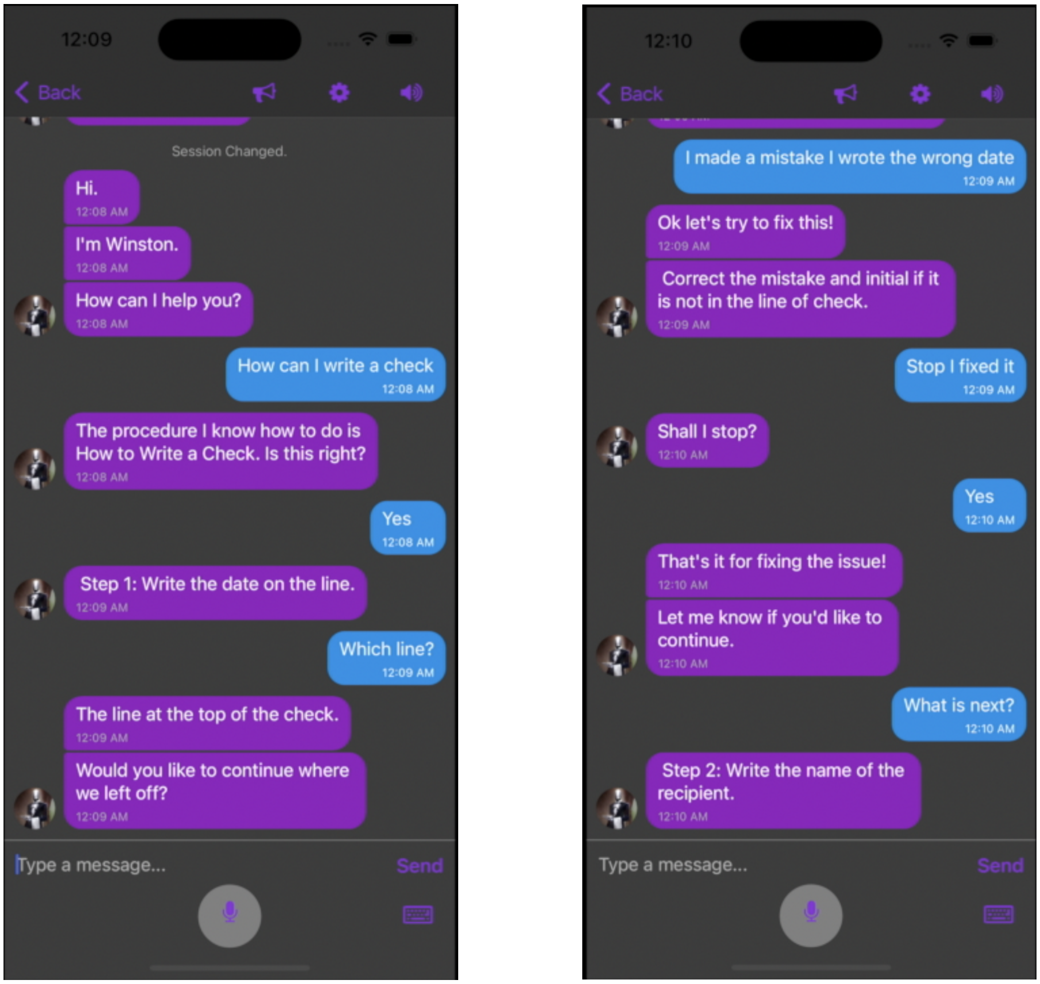
.

conversation. The backend returns with the initial turn(s) from the agent indicating availability for help. The agent's responses are shown in text bubbles and through audio if the user has it enabled. Audio can be enabled or disabled by pressing the sound button in the top right of the screen. For speech-to-text, on the native version of the apps (iOS and Android) we utilize the react-native-voice package, which is a wrapper around the device's native dictation functionality. In the web version of the app, we utilize the react-native-webrtc-web-shim[3] package which permits the use of streaming sound which is sent to the IBM Watson Speech-to-Text Streaming API and returns transcriptions on the fly.

The interface was designed using react-native in order for it to work over a variety of devices and screens, while maintaining the core functionality of the interaction. Because of this, the interface works in most browsers on most devices, and it can be installed as a native app on Android and iOS. We utilize the Expo library to host the

---

[3] https://github.com/react-native-webrtc/react-native-webrtc-web-shim



Hi.
12:08 AM

I'm Winston.
12:08 AM

How can I help you?
12:08 AM

How can I write a check
12:08 AM

The procedure I know how to do is How to Write a Check. Is this right?
12:08 AM

Yes
12:08 AM

Step 1: Write the date on the line.
12:09 AM

Which line?
12:09 AM

The line at the top of the check.
12:09 AM

Would you like to continue where we left off?
12:09 AM

I made a mistake I wrote the wrong date
12:09 AM

Ok let's try to fix this!
12:09 AM

Correct the mistake and initial if it is not in the line of check.
12:09 AM

Stop I fixed it
12:09 AM

Shall I stop?
12:10 AM

Yes
12:10 AM

That's it for fixing the issue!
12:10 AM

Let me know if you'd like to continue.
12:10 AM

What is next?
12:10 AM

Step 2: Write the name of the recipient.
12:10 AM

Figure 6-3: In the chat screen, after logging in, is where a user would communicate with the agent. Here, a user can either dictate (by pressing the microphone button) or type their responses to the agent. The user can also mute the agent by pressing the sound button at the top right

.

web version of the app and to make builds that can be downloaded and installed on iOS and Android devices.

With the agents that utilize the IBM Watson Assistant in the backend, the GUI pings the Assistant 3 times for the assistant to introduce itself and leave an open question of what can it help with. This follows the self-identification (Agent) C1.1 pattern along with an offer of help. We now give details on the backend server that the GUI communicates with.

Back-end Web Server

The backend server which interacts with the GUI for the back and forth conversational exchanges was developed utilizing Python and the Flask library. The server implements a variety of endpoints. These endpoints serve as abstractions for things such as, generating contextual procedure knowledge graphs, generating steps from knowledge graphs, performing web searches for unseen procedures, querying large language models, and querying the Watson API for Assistant, transcription, and speech generation capabilities.

Performing Web Searches for Unknown Procedures To handle unknown procedures, we utilize the process described in Section 4.7.3. We rely on the IBM Watson assistant to successfully extract what the user desires to learn to do and pass it into our server to kick-start the search, extraction, and generation process.

Generating Contextual Procedure Knowledge Graphs & Generating Steps Now with the steps that we have extracted, we can run the procedure that we describe in Section 4.7.4, to generate a contextual knowledge graph, reason through it, and generate steps as we describe throughout Chapter 5. Briefly, the process is the following:

1. For a given procedure, we iterate through all of its steps.
2. We extract any noun phrases and keywords that are found in the step
3. We generate, using hints, contextual assertions (specific), for permutations of the noun phrases as subjects, and specifying the relations types
4. We collect the resulting assertions as a KG of the given step
5. After we have generated a KG for every one of the steps, we pass this graph to our Knowledge Driven Procedural Step Generator
6. The generator takes the top-rated K of those facts and hands them off to the reasoner, to return a subset of at most 64 of the top-rated facts

7. We then pass this subset into the realizer to get a procedural step

In the case that an exception occurs, we once more utilize the process in Section 5.4 to handle the exception.

Querying Large Language Models We query a Large Language Model to help with reasoning, step extraction, and to serve as a baseline. In our server, we utilize the OpenAI API Python package to do this. For our conversational agent, and for the step extraction, we utilize the ChatGPT API with a prompt that we describe in Section 6.5. We note that although our backend queries the OpenAI Models, this could be replaced with other more efficient/open-sourced models such as LlaMA [170] or PaLM [25]. Additionally, we note that the components that utilize these LLMs, can be replaced in their entirety with simpler components (e.g., reasoner can be replaced with some reasoning agent, step extraction can be replaced with a smaller LM that gives spans, and the baseline conversational agent can be replaced with a procedure lookup system similar to the baseline that we utilize in our tests. We utilize this LLM querying functionality to handle user inquiries whenever they come up with our intent-based agent. If possible, we give the LLM the context of the procedure we are in along with the user's question and request and answer from the LLM.

Watson API We now describe the APIs that we access from IBM to, build a basic conversational agent, convert speech to text for inputs, and convert text to speech for responses. We note that we utilized IBM services for our work, as we had access to a reference implementation for conversational patterns, but any other service such as DialogFlow from Google and any other speech-to-text/text-to-speech could be utilized.

IBM Watson Assistant

IBM Watson Assistant is a chatbot development platform that allows businesses to create conversational interfaces for a variety of user interactions. It uses natural language processing (NLP) and machine learning (ML) technologies to understand user queries and respond with relevant and accurate information. Watson Assistant can be integrated with various messaging platforms and can also be trained to recognize customer intents and personalize responses.

In our work, for every user, we start a Watson Assistant session in which the system can receive inputs and provide outputs. We combine the Watson Assistant with our server by leveraging web-hooks from the service. These hooks are just REST requests into our server to query for procedures and instructions.

IBM Speech-to-Text

IBM Speech-to-Text is a cloud-based service that converts audio and voice into written text. It uses advanced machine learning algorithms to recognize and transcribe speech in real-time, with support for a variety of languages and accents.

In our work, we utilize this speech-to-text functionality to develop a streaming transcription system. When a user presses the microphone button, seen in Figure 6-3, the audio streams to the Watson service, which provides partial and final transcriptions. The partial transcriptions are displayed as they come in from the service, and when the final transcription comes in, it replaces the partial ones.

IBM Text-to-Speech

IBM Text-to-Speech is a cloud-based service that converts written text into natural-sounding speech. It uses deep learning neural networks to generate speech that sounds more human-like than traditional text-to-speech engines. This service supports a wide range of languages and voices, allowing customization of the voice and tone of the text-to-speech output.

We utilize this system to convert agent text responses to voice responses. We use the "US Allison" voice in our system, but note that this can be customized.

Registered Users and Interactions Indices These two indices serve to store users and their encrypted passwords, along with any interactions that they may have had with an agent. We store the username, the password (encrypted in Base64 for simplicity) in the registered user index. In the User Interactions index we store a timestamp of the interaction, the content of the interaction, who initiated the interaction (whether it is a message from our agents or from a user), a Session ID, and a User ID. We don't list tables with descriptions and examples for these for brevity.

6.4.3 Intent-Based Procedural Guidance

We incorporate a reference version of the conversational patterns described in Section 6.2. We implement these patterns in the Watson Assistant. This conversational agent uses intents to determine what a user is saying and wants. Some examples of intents are: requesting instructions, affirmations and disaffirmations, continuers, definitions requests, among others. We utilize over 80 intents in our agent. Intents also serve to perform slot-filling to extract things such as the procedure that the person wants to do or the error that they encountered. The Watson Assistant utilizes a dialogue tree that, after intent classification, the dialogue tree is navigated and dialogue nodes are evaluated to determine what is the agent's next action. The dialogue nodes serve as the implementation of the conversational patterns. The Watson Assistant also provides functionality to perform web-hooks in dialogue nodes. web-hooks serve to call remote resources such as our server with the gold procedures and with the dynamic procedures.

To provide some flexibility in our agent, we implement definition requests and general and procedure question answering utilizing large language models. The reason for this is that users may provide contextual questions that it may be hard to pre-script the answers for. One example is, when writing a check, our procedural guidance agent may give the instruction: "Write the amount in numbers", and a user may respond to that by saying: "Where do I do that?". If we had to pre-script answers, we would have to determine what possible values would "that" refer to, within the context of the procedure that we are guiding on. We found that a simpler way to do this would be to, utilizing a Webhook, call a large language model, and prompt it for an answer. In the case that a person is being guided in a procedure, we can give the large language model enough context (the procedure name, the person's question, the last step given) for it to come up with a reasonable answer. In the case that we are not in a procedure, we rely on the large language model's capacity to answer open-domain questions. We note that in the way our system is implemented, there is no option for barging-in [164]. What this means is that even if users speak while the agent is talking, the agent will not pause/respond to users barging in. Future work should investigate incorporating barging in behavior for procedural guidance, as this may give users the chance to interrupt and ask for clarification for an agent. We also note that it is possible, although we did not implement it, to add a timer after the last utterance by the agent. If this timer completes without any user interaction, the system should either continue guidance or ask if the user is still there.

Gold-Lookup Agent

For our agent that gives gold instructions, we utilize web-hooks to perform a large-scale search for procedures in our collection. This search is performed on the "Content" field of the procedure documents that we have saved in ElasticSearch, and it is done with the FAISS library with embeddings produced from the Sentence-transformers library and the model "all-mpnet-base-v2" which currently has the strongest performance in a variety of semantic tasks[4].

After a procedure is retrieved, a user is questioned on whether that is the correct procedure or not, and if the user disaffirms, the agent asks the user to rephrase the procedure. If a user affirms the procedure, then the agent begins the extended telling with the instructions from the ingested procedure. If an error occurs, a user needs to notify the system that the error has occurred, and give details of the error. When this happens, another search is performed, and the closest matching procedure is then utilized for a recursive extended telling.

Knowledge Driven Dynamic Agent

For our dynamic agent, we utilize a similar strategy as the gold-lookup agent. The only difference is that steps are generated dynamically from the contextual knowledge graph that is derived from the procedure. We keep the procedural search to be able to easily determine if we have the knowledge/information necessary to give guidance

---

[4] `https://www.sbert.net/docs/pretrained_models.html#sentence-embedding-models/`

on a topic. In contrast to the gold-lookup agent, our dynamic agent also implements a web-search and LLM powered extraction to find out how to do procedures that are not in our collection. This same process can be used for the gold-lookup agent, but we have no way to enforce that the extracted steps are useful, whereas with our Knowledge Driven agent, we can re-synthesize the procedure from knowledge which may add a layer of verification, although future work should explore this more in-depth.

In the case that an error occurs, we perform a call into our server to, look up the graph of the current procedure, look up/generate a graph for an error-fixing procedure, and combine the knowledge from the last step that was performed with the initial knowledge of the steps that are to be performed. The reason for the knowledge combination is to give context to the error-fixing procedure of what was being done before.

6.5 Large Language Model-Based Guidance

6.5.1 Plain, zero-shot guidance agent

In addition to providing guidance with an intent-based agent, we developed a LLM-based agent to evaluate for procedural guidance. Large language models (LLMs) (i.e., models that have 10B parameters or more) have recently been shown to perform incredibly in a multitude of zero-shot scenarios [192]. In particular, we look at the more popular GPT-3 [20] and ChatGPT. These models have appeared in the news in many use cases, but in our work, we explore utilizing them for procedural guidance. We explore how well these models fare when prompted simply to give guidance and to give it in a step-by-step manner. From initial tests, these models perform incredibly well to recall procedures that they are prompted for, and in the case of any error guidance, they seem to be able to take context into consideration when responding. However, as we will see later on in Chapter 8, these models are not perfect. We discuss later some of the shortcomings of these models and how they could be addressed in future work.

Initial Prompt

In our work, we utilize the following prompt to guide conversations that are powered by the LLM:

> *You are in a conversation with another person. You must always answer in one sentence. If you need to provide guidance, provide it one step at a time. Your name is Winston and you are here to help. You always introduce yourself at the beginning of a conversation.*

We find that with this, it is enough to converse with users on procedural guidance topics.

6.5.2 Hybrid: Knowledge Driven LLM

As a final system, we developed a hybrid of the knowledge driven procedural step generation system, and a LLM (ChatGPT-based) system. This hybrid system tries to leverage the advantages of very large language models, such as robust context, understanding, and question answering capabilities, with the steps and knowledge from our graph-based step generation. An advantage of our graph-based system is that it can come up with procedures dynamically from up-to-date sources and generate steps for them. Additionally, In contrast, and as will be shown in Section 7.4, LLMs have knowledge of the world up until the point in time that they are trained in. What this means is that on domains that change (e.g., operating systems and software with updates), the knowledge of how to perform tasks needs to be updated or refreshed in some way. Although this addresses coming up with unseen procedures, for mission-critical domains such as medicine, more care needs to be taken in getting up to date and referable information. We essentially replace the complete conversational system that we developed in Watson with LLM prompts. However, when we do this, we still need a procedure name to guide our step generation system.

LLM-based intent recognition and slot filling

We utilize the LLM system with the prompt:

> (While giving instructions to share duration on iPhone)
>
> Would a person require guidance if they said this (Respond with Answer(Yes/No) and Procedure Name)?
>
> While sharing my location, I made a mistake, I need to change duration of the share"
>
> Answer(Yes/No):
>
> Yes
>
> Procedure Name:
>
> Change duration of location sharing on an iPhone
>
> Would a person require guidance if they said this (Respond with Answer(Yes/No) and Procedure Name)?
>
> Hi Winston
>
> Answer(Yes/No):
>
> No
>
> Procedure Name:
>
> None
>
> Would a person require guidance if they said this (Respond with Answer(Yes/No) and Procedure Name. Respond with a No and Procedure Name: None if it is some kind of clarification or request for breaking down a process)?
>
> [User Input]

Where “[User Input]” is the user’s last utterance. In addition, since we are using the ChatGPT system as our LLM, we can send each example question as a system question with an agent answer. We give these examples so that the system will be consistent in its formatting, and we can parse whether it is a request for a procedure and what.

We do notice, however, that the system sometimes misclassifies continuers and clarifications as procedures in themselves, sometimes throwing off the KDPSG system. However, this usually occurs after the user has requested a procedure, and the result of this is ignored in future turns.

When this simple procedural intent classification returns true, and the name of the intent is returned, the KDPSG system is called to generate knowledge and steps for the user's procedure

KDPSG+LLM

Now, after the KDPSG system returns, it returns with a set of sets of assertions and with the steps that correspond to those sets. We then sample from each set of assertions, the top 256/number of steps. We divide 256 between the number of steps, because from some testing, more than this amount, in combination with the steps, would overflow the context for the LLM. We then combine them into one list along with the steps from the KDPSG system and supply them in a prompt:

> "If asked for instructions on how to do something use the following pattern to convey the instructions: Confirm if needed what the person wants to get guidance on, begin the interaction by providing one step at a time, never give a full list of steps unless the person asks for it wait for a person to confirm or acknowledge a step before providing the next one. If you are asked a question about the procedure, you will try to answer it, but remind the user that you should continue once you are done, and that your information may be outdated. If you are giving instructions on a procedure, and a person makes a mistake that cannot be addressed with a question, you will guide them on how to fix the issue with the same pattern as above. When the person finishes the procedure you thank them and ask if there is anything else that you can help with. You also start the conversation with is there anything I can help you with. If you are

> responding with a procedure about [PROCEDURE NAME] incorporate these facts if possible:
>
> [FACT LIST]
>
> Use the following steps if needed (always correct grammar errors):
>
> [KDPSG STEPS].
>
> Make sure to paraphrase the steps to make them easy to understand."

We substitute [PROCEDURE NAME] with the inferred procedure name from the slot filling from above, [FACT LIST] with the list of facts of the procedure, and [KDPSG STEPS] with the knowledge driven steps. What we observed is that the LLM tends to utilize these steps in its instruction giving. We also explored giving a description of the extended-telling pattern similar to Section 8.4. We also observed that the LLM does not always use the description of the conversational pattern, and that it does not always use the knowledge/steps that are given to it.

6.6 Summary

In this sixth Chapter, we present the conversational agent architecture, a web-server backend and GUI frontend, that we will utilize in the next Chapter for user testing. Our primary agent consists of an intent-based system that incorporates conversational patterns to convey procedural instructions. We incorporate a small extension to the extended telling conversational pattern to be able to handle longer repairs. We additionally present a LLM powered agent that we prompt to provide guidance for people, and a hybrid Knowledge Driven LLM to see if we can make up for the later discovered usability faults of the intent-based systems. With all of this, we are now ready to try a conversational agent for procedures on people.

Chapter 7

Evaluating a Conversational Agent for Procedures

> *My point is to take user interface evaluation with a grain of salt. Let's continue to do, and to encourage doing, user testing, but let's keep in mind that the results will not be definitive and need to be understood and interpreted in depth. Let's not reject out of hand new interface innovations that have not been tested or don't immediately show spectacular results in testing. Let's not let knee-jerk "tests have shown that" replace detailed analysis. We need more judgment in evaluation of user interfaces, not just more calculation. -Henry Lieberman*

7.1 Overview

In the past two chapters, we were able to generate contextual knowledge graphs that stem from procedures. We were also able to ingest these graphs, using a simple metric, or a large language model powered reasoner, and filter them to produce procedural steps. We also described how to wrap this procedural step generation system in a conversational agent. We now evaluate this agent in a variety of studies to see its effectiveness and its usability. Additionally, we test a hybrid approach to get the best of both worlds: the robustness of LLM based agents with the factuality of our knowledge-based approach.

7.2 Evaluation 1: Task Completion

7.2.1 Overview

We conducted a study to evaluate the effectiveness of our conversational agent in guiding users through procedures. The study involved asking users to perform a smartphone related task by asking for and receiving guidance from one of three agents: a gold-instruction retrieval system, a knowledge-driven procedural step generation agent (KDPSG), and a ChatGPT-based agent (GPT).[1] Users were then asked to report back on whether they were able to successfully complete the task and give feedback on the task. Throughout the chapter, we refer to these agents as our baseline, KDPSG, and GPT, respectively. Our findings show that all three agents *are capable of* guiding most people (70%+ of participants) through procedures. Our findings also suggest that our two intent-based agents need improvements in usability, particularly in the intent-matching and slot-filling capabilities. Lastly, we find that LLMs are very capable of, in a zero-shot setting, providing guidance in most procedures, although their information may be out-of-date.

7.2.2 Research Questions

The explicit questions that we set out to solve with this experiment are the following:

1. H1: Can a conversational agent with conversational patterns guide people through procedures?
2. H2: Can generated instructions be as effective as gold-standard ones in procedural guidance?
3. H3: Is this agent natural and conversational?
4. H4: Can we get some design feedback on our procedural guidance agent?

---

[1] These are the agents that we describe in Chapter 6.

7.2.3 Study Design

To assess the effectiveness of our guidance, we devised a set of tasks that individuals may encounter in their day-to-day lives, with a particular focus on technology. Our ultimate goal is to develop an agent that can provide guidance on any task, and we designed these tasks with the intention of making them unusual enough that users may not be familiar with them, but accessible enough that a wide audience could be capable of performing them. This ensured that individuals would have to rely heavily on the voice assistant for guidance. The tasks that we selected for this study are as follows:

1. Mirror a smartphone to a display wirelessly
2. Share your location through text messages on a smartphone
3. Compose an email in your smartphone with an attachment/embedded image
4. Organize smartphone applications into folders
5. Compose a Multi-media note in your smartphone
6. Create a calendar entry in your smartphone with a video conference link
7. Set up an automation on your smartphone
8. Set up an email signature on your smartphone

We aimed to keep our study as varied as possible, so we did not specify a particular type of smartphone for the tasks. However, we were able to leverage gold-standard procedures for the tasks in our backend. These are procedures that are written by the manufacturer or a reputable entity, ensuring that they are up-to-date and reliable, as the writers have in-depth knowledge on the topic. In particular, we made sure to have the most up-to-date version of these procedures for both iPhone and Android users.

In our work, we opted to perform this study with the Prolific[2] platform. Prolific is an online platform that researchers use to find people who are willing to participate

---

[2] `https://www.prolific.co/`

in their studies. It is a widely used platform because it offers a large pool of participants who can be screened based on specific criteria. The platform allows researchers to create and publish their studies, set eligibility requirements, and compensate participants for their involvement. Additionally, participants in Prolific are vetted, and fair pay is enforced, which typically results in higher quality responses than other platforms.

In Chapter 6, we created an agent that could be accessed through a web application available to anyone. To enable remote participants to use a specific agent in our study, we used URL query parameters to generate links that specified a certain agent for participants to access.

To test the effectiveness of our agents on a set of tasks, we devised a strategy using a within-subjects design [13] to maximize the population of the study. First, we asked users to rate their frequency of performing the tasks on a Likert scale of 1-5, with 1 indicating they had not performed the task in the past month and 5 indicating they performed the task almost daily. From the tasks that users had performed the least, we randomly selected one and had the user ask for guidance on how to perform the task using a voice agent. We had the user perform the task twice and provide feedback each time, with the guidance of a different agent. We conducted the study in blocks of two, with the baseline agent compared against another agent, which may be the GPT agent or our KDPSG. We note that we indicated that users should try to utilize the microphone as much as possible, but that they were free to utilize the keyboard and write to the agent if there were issues.

In addition to the feedback form, the users had to fill out a pre-study questionnaire which consisted of the following questions:

- How familiar are you with conversational agents (e.g. voice assistants like Siri, Alexa, Google Assistant)?
- What is your perception on the usefulness of conversational agents (e.g., Siri, Alexa, Google Assistant)?
- In a typical day, how many times do you interact with a conversational agent

(e.g., Siri, Alexa, Google Assistant)?

and a post-study questionnaire that consisted of these questions:

- What is your perception on the usefulness of conversational agents (e.g., Siri, Alexa, Google Assistant)?
- Do you think the tasks given were appropriate for this kind of agent? Why or why not?
- Anything else? Is there any other feedback that you would like to give? This can be on the survey, the interactions, the interface, etc.

Finally, after every procedural interaction with a specified agent, a participant has to answer the following questions:

1. I felt the agent was conversational (5 point Likert scale)
2. I felt the conversation was natural (5 point Likert scale)
3. I had to pay special attention regarding my phrasing when communicating with the agent (5 point Likert scale)
4. I find that the agent understands what I want and helps me achieve my goal (5 point Likert scale)
5. The agent gives me the appropriate amount of information (5 point Likert scale)
6. The agent gave relevant information during the whole conversation (5 point Likert scale)
7. I thought the agent was easy to use (5 point Likert scale)
8. I would imagine that most people would learn to use this agent very quickly (5 point Likert scale)
9. I found the agent unnecessarily complex (5 point Likert scale)
10. I think that I would like to use this agent frequently (5 point Likert scale)
11. Did you succeed in your task (what the agent guided you on)? (Binary Yes/No)
12. Did you accomplish your goal (what you were told to do)? (Binary Yes/No)

13. Did the agent misunderstand you? If so, how? Otherwise, write N/A. (Open response)
14. Did the interaction with the agent frustrate you? If so, how? Otherwise, write N/A. (Open response)
15. Did you encounter any problems when interacting with the agent? Otherwise, write N/A. (Open response)
16. Did you encounter any problems when performing your task? If so, what? Otherwise, write N/A. (Open response)
17. What did you feel worked very well when completing the task? (Open response)
18. What did you feel did not work well when completing the task? (Open response)
19. What did you feel was useful or effective when interacting with the agent? (Open response)
20. What did you feel was not useful or effective when interacting with the agent? (Open response)
21. Anything else? How could this interaction be improved? (Open response)

These may seem like many questions, but they cover the research questions that we wanted in both a qualitative (open-response) and quantitative (Likert scale) manner. We sample some of the general usability questions for conversational agents from [15]. The complete survey can be seen in Appendix A.1.

During the study, we assigned each participant a unique random identifier that was used to create an account in our backend system to keep a record of their interactions. We captured the conversation turns between the participant and the system, as well as the system's responses, and associated them with a unique session identifier and timestamp. By doing so, we were able to track which agent the participant was using during the session and order by time the sessions. We note that through the Prolific platform, we recruited participants with the following demographics:

- Participants from the US, and the UK.

- A balanced sample of genders
- Minimum Age of 18 and a Maximum Age of 55
- Technology use at work: not at all, less than once a week, about once a week, 2 or 3 times a week, 4 or 6 times a week, about once a day, more than once a day
- Weekly device usage: Never, Once a week, 2-6 times a week, Every day, Multiple times every day
- Exclude participants from previous studies

With these demographics, we could somewhat guarantee that we would have a mostly English speaking, maybe experienced or inexperienced with technology use participant. We selected participants between the ages of 18-55, because this is where currently most of the usage market for voice assistants lies[139], [citepewresearch. Additionally, below and above those ages the participants would be considered a vulnerable population, and we may have to do some accessibility changes to our work, which given time constraints was not possible.

7.2.4 Study Results

Golden Instruction Retrieval vs. KDPSG

We had 30 participants in our study (N=30). We now go through the results of our survey. We find that participants took on average 27.56 minutes to complete the survey. We find that most (N=28 or 93%) of the participants are moderately or more familiar with conversational agents, and that a minority of participants are slightly familiar with them. None of the users were unfamiliar of conversational agents. A view of this distribution can be seen in Figure 7-1. We also saw from the pre-study questions that most people (86%) find conversational agents to be moderately useful or more, as seen in Figure 7-2. Lastly, we see that most participants interact with a conversational agent at least once a day (83%) as seen in Figure 7-3.

Now, in Figure 7-4 we report the reported performance of tasks.

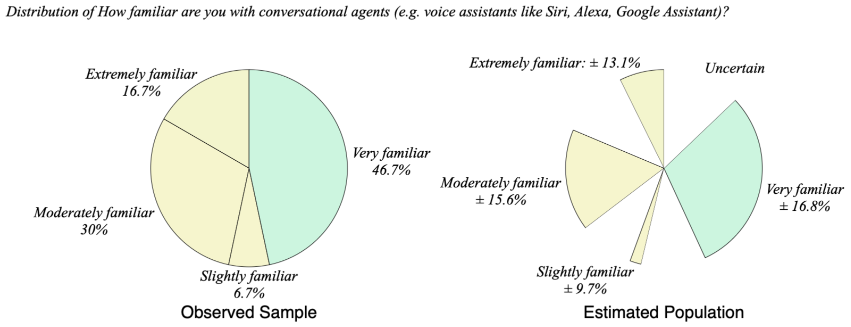


Figure 7-1: User distribution of familiarity with conversational agents. We see most people are at least moderately familiar with conversational agents.

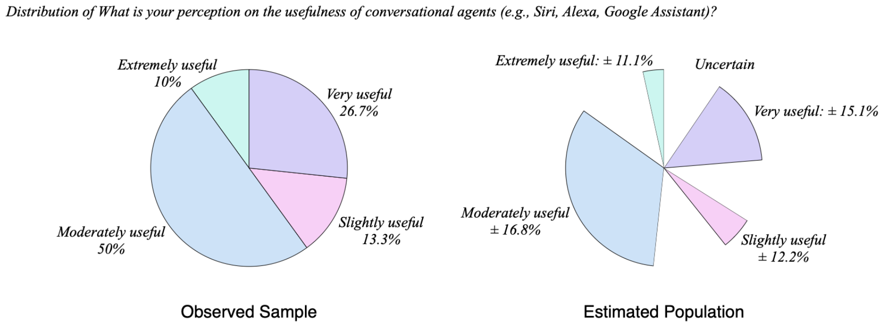


Figure 7-2: Distribution of participant's perception of general usefulness of conversational agents. We see that most people find the agents useful.

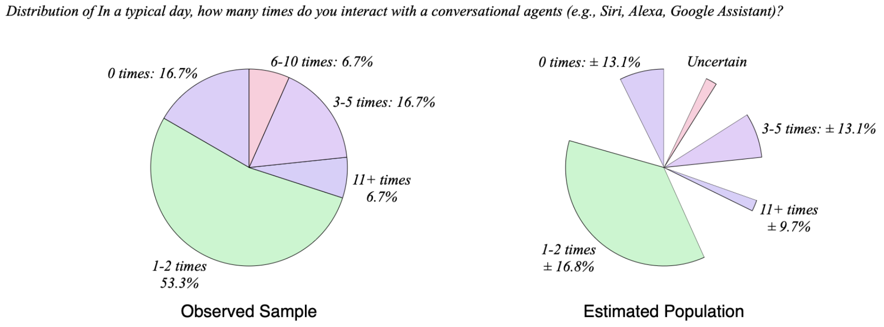


Figure 7-3: Distribution of participants reported interaction quantities with conversational agents. We see that more than half of people use an agent at least once a day or more.

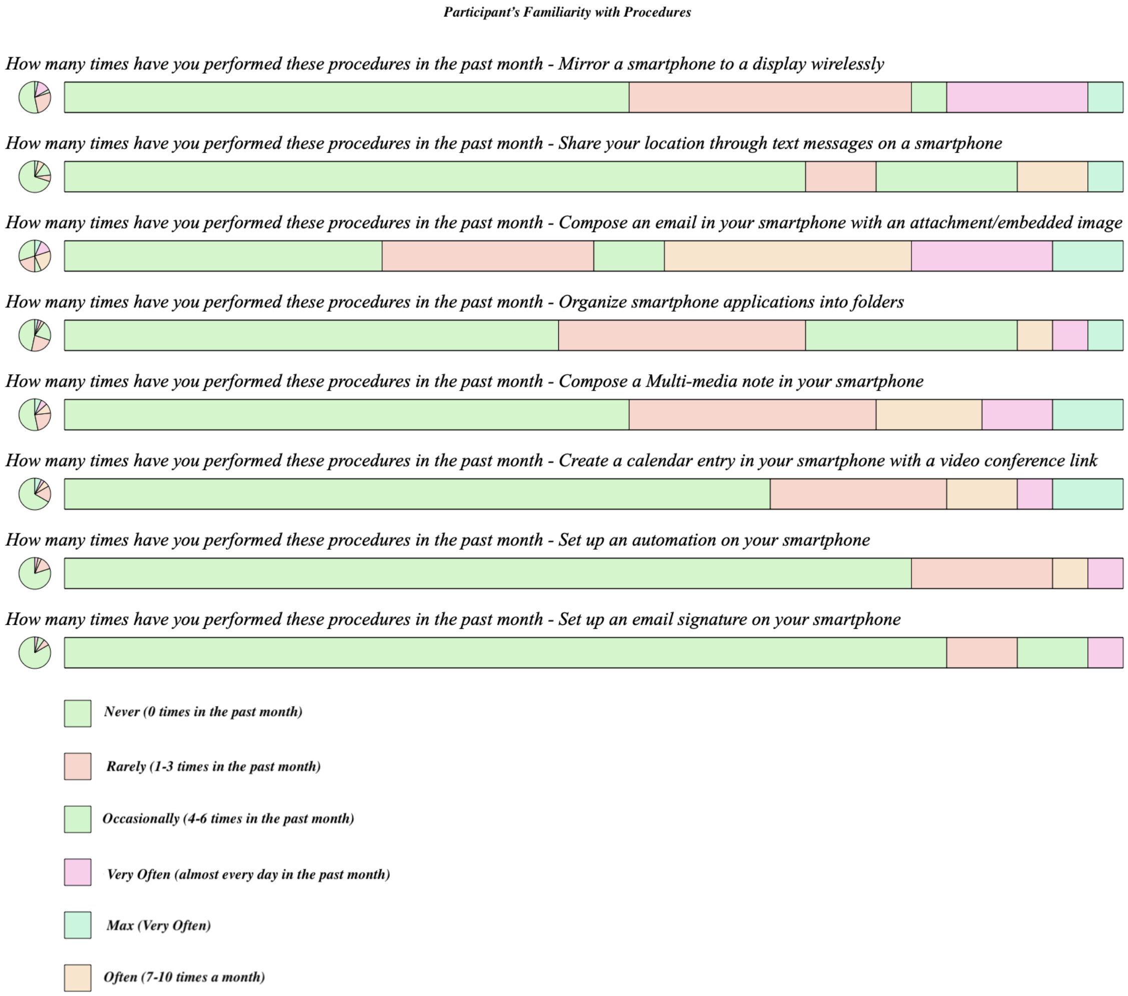


Figure 7-4: Distribution of participants reported monthly performance of tasks. We see that almost all the selected procedures fell into the "rarely" category (value=1), with only the composing an email procedure being the one that people perform the most.

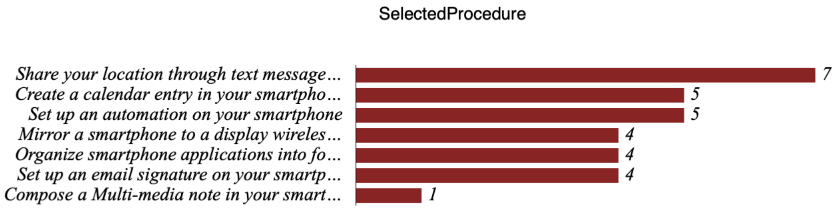


Figure 7-5: Distribution of the procedures that were assigned to participants. We see that this follows an inverse distribution as the task ranking, where the procedures that were ranked as most familiar are the least that we select for participants.

We see that most tasks, save for composing an email with an embedded image, have been performed "rarely (1-3 times)" or less in the past month. This is means that our selection of tasks was good, with 7/8 tasks that people do not perform frequently. From this reported performance, we then sampled 1 task randomly from the set of least performed tasks. The distribution of tasks that a participant was given can be seen in Figure 7-5

Then, we reviewed the feedback of whether the tasks that were given were appropriate or not for a conversational agent, out of the 30 feedback statements provided, 19 of them were positive, saying that the tasks were appropriate for conversational agents, 8 were negative saying that this agent cannot be used for these tasks, and 3 were neutral. The most common positive feedback was that the task was simple and appropriate for an agent, while the most common negative feedback was that the agent struggled to understand the user and therefore could not be used for such a task unless it was developed further. Several participants suggested that the agent would be more useful for simple tasks, while others mentioned that they would still prefer to use Google for instructions. Overall, while there were some concerns about the agent's ability to understand tasks, most participants felt that the task that was picked for them, was appropriate for such a conversational agent and had potential for future development.

Now we move on to the more important results from the study. The most important results can be seen in Figures 7-6 and 7-7. From here, we can see that a significant portion of ($p{=}0.005$ with a binomial test assuming a 50% hypothetical success rate) participants (22/30 interactions in both the baseline agent and the knowledge driven

Step Number	Gold Procedure Step	Knowledge Driven Step
1	Open the notes app	Step 1: Open the Notes app to make notes on multi-media instead of paper
2	Tap the compose icon	Step 2: Tap the compose icon to make sure it is in multi-media.
3	Tap on the camera icon and choose a photo or video from your photo library, or take a new photo or video	Step 3: Tap the camera icon to choose a new photo or video to take from your photo library
4	Use the other icons to add different media like bulleted list	Step 4: Add other media like bulleted list or format to use different media

Table 7.1: Side-by-side example of golden instructions vs. procedural instructions. We note that this procedure was unseen by the knowledge driven system.

agent) were able to complete the task that they were assigned, independent of the agent, and that they were able to follow the instructions that an agent gave them. We note that there is no significant difference (p=1.0 using a Fisher exact test, and a χ^2 test) between the baseline and the knowledge driven approach in regard to task completion (what the agent guided the participant in) and goal completion (what the participant was told to do). This indicates that the instructions that were given were just as effective as the gold standard ones. . We now give an example of a generated procedure vs. a gold-standard one for the procedure of “Compose a Multi-media note in your iPhone” in Table 7.1.

We can see that our model is able to reconstruct the procedure somewhat effectively, although it is not perfect. This is similar to the automated metrics that we have obtained in Chapter 5, which say that there is a high overlap between what the step contains and what we generate, but it is not perfect. We note that the reconstruction is based on having an aligned set of facts and steps, hence the order of the steps is similar. This is a limitation that is acknowledged further on and could be resolved by using retrieval methods which learn to automatically align facts and steps. However, we leave this as future work.

From this, we can gather that the instructions that the agent gave were useful enough, that most people with both agents, (upwards of 73% of the interactions (44/60), or respectively 22/30 for each agent) were able to accomplish the task they were assigned. For the KDPSG agent, 6/8 interactions failed because of intent matching/slot filling issues, and the other 2 were because of other issues. We note that in roughly 27% of the interactions (16), participants were unable to complete their tasks. When we look at the percentage that struggled to complete the task, from the feed-

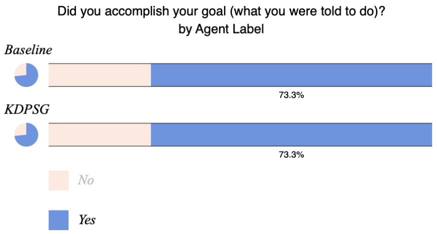


Figure 7-6: Distribution of whether a participant was able to accomplish the goal they were given or not. We see that most of the participants succeeded in this.

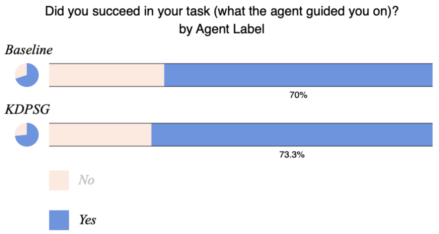
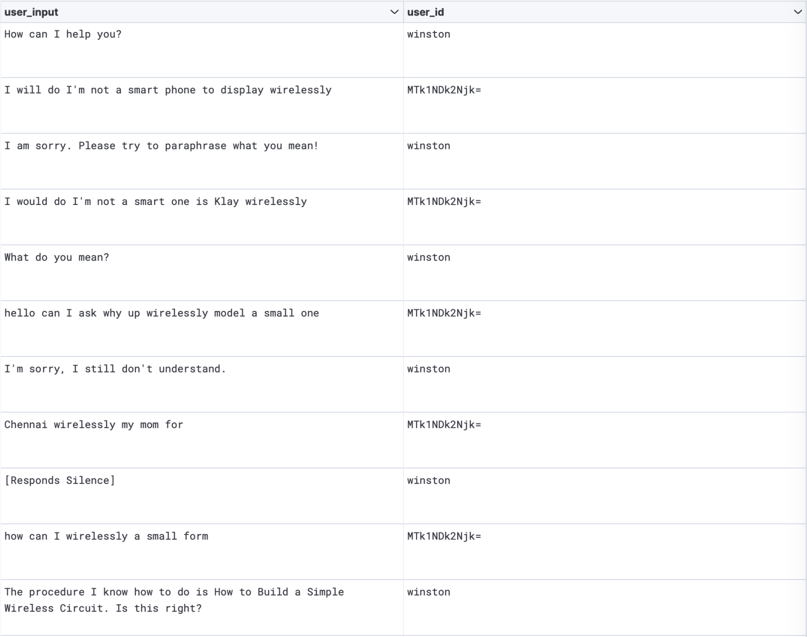


Figure 7-7: Distribution of whether a participant was able to accomplish the instructions that the agent gives. We see that most participants were able to follow what the assistant guided them on.

back that they give along with the interactions that were recorded, multiple users experienced difficulties using the assistants due to various issues, including accent, microphone problems, and difficulty in phrasing, and the agent understanding, their questions. The users note that the assistants consistently misunderstood them, transcribing incorrect sentences, and asking them to repeat or paraphrase their requests. In some cases, the assistant would not understand specific queries. Lastly, the assistants often asked users to clarify their requests.

We observed that some people *only* utilized the microphone to interact with the agent. Although this would bring more natural responses, there were some cases, at least 3/60 (5%), in which the speech to text failed catastrophically and blocked the user from completing the task. When we look at this in the case of failures to complete the task, these 3 transcription errors account for more than 18% of the failures (3/16) One such interaction can be seen in Figure 7-8

For other interactions that fail (8 for each agent), we find that most users were unable to either start the extended telling dialog tree due to a misunderstood intent (8/16), or users were unable to find the procedure that they needed to do with the semantic matcher(2/16), which was because of very vague phrasing or problematic slot filling in the intent. We looked through the successes when completing the tasks, and found that these participants sometimes rephrased their requests in such a way that the agent would understand. In Figure 7-9 we can see how a successful attempt



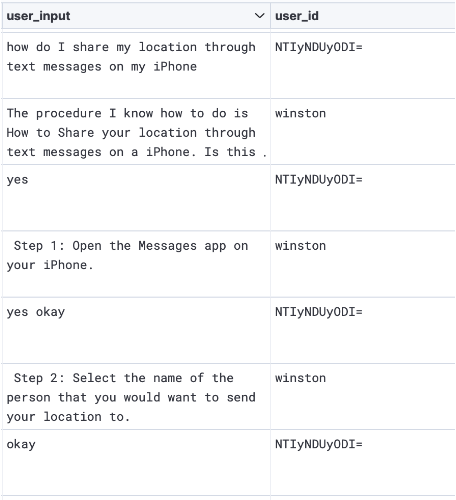
user_input	user_id
How can I help you?	winston
I will do I'm not a smart phone to display wirelessly	MTk1NDk2Njk=
I am sorry. Please try to paraphrase what you mean!	winston
I would do I'm not a smart one is Klay wirelessly	MTk1NDk2Njk=
What do you mean?	winston
hello can I ask why up wirelessly model a small one	MTk1NDk2Njk=
I'm sorry, I still don't understand.	winston
Chennai wirelessly my mom for	MTk1NDk2Njk=
[Responds Silence]	winston
how can I wirelessly a small form	MTk1NDk2Njk=
The procedure I know how to do is How to Build a Simple Wireless Circuit. Is this right?	winston

Figure 7-8: Example of a catastrophic transcription error that blocked a user.

looked like, in contrast to an intent matching failure in Figure 7-10.

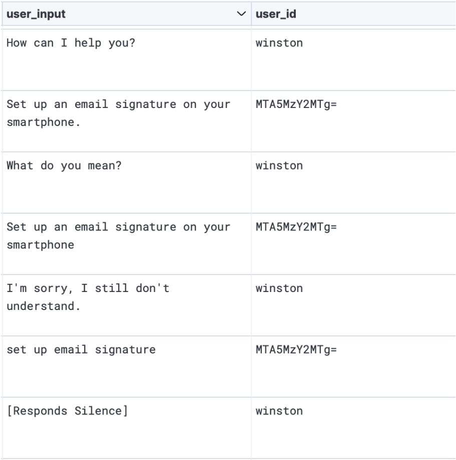
Now after analyzing the failures and successes, we can see more clearly that the people that were able to achieve the guidance, were able to navigate the intent matching and slot filling successfully, whereas many of the people that failed the task were not. We also saw that people had to paraphrase many times to try to succeed in the intent matching. This need to paraphrase multiple times is likely to have influenced people's perception of the usability of the system negatively. Since we utilized the same agent for dialogue management, it is unsurprising to see that the flaws that were found in the baseline retrieval method were also found in our KDPSG agent.

In all the usability questions, there were no significant differences in the agents ($p>0.05$ with a Wilcoxon Signed Rank Test). We first begin with Figure 7-11 which shows that people were neutral on whether the amount of information that was given to them was appropriate when interacting with the agent. We move on to Figure 7-12 which shows that people did not find the agent to be conversational. We observed from the feedback given in the open responses, and it is more evident in Figure 7-16 that people had to be very careful with the phrasing that they used with the agent; they had to be very specific in the procedure that they asked for. In the open response



user_input	user_id
how do I share my location through text messages on my iPhone	NTIyNDUyODI=
The procedure I know how to do is How to Share your location through text messages on a iPhone. Is this .	winston
yes	NTIyNDUyODI=
Step 1: Open the Messages app on your iPhone.	winston
yes okay	NTIyNDUyODI=
Step 2: Select the name of the person that you would want to send your location to.	winston
okay	NTIyNDUyODI=

Figure 7-9: Example of successful intent matching and guidance.



user_input	user_id
How can I help you?	winston
Set up an email signature on your smartphone.	MTA5MzY2MTg=
What do you mean?	winston
Set up an email signature on your smartphone	MTA5MzY2MTg=
I'm sorry, I still don't understand.	winston
set up email signature	MTA5MzY2MTg=
[Responds Silence]	winston

Figure 7-10: Example of failed intent matching. Examples originally did not include the phrasing "set up [PROCEDURE NAME]".

feedbacks of "Anything else? How can this interaction be improved", many people (15) said that the agent needs some work in "understanding" better what the user said. All of this then leads to the other results that participants did not think that the agent was easy to use (Figure 7-13), that people would not use the agent frequently (Figure 7-14), and that they did not think that the conversation felt natural (Figure 7-15). All of these results, suggest that there needs to be more engineering work in improving the intent-recognition system, and that it adversely affects people's perception of the usability of the agents that they tested.

Finally, we conclude with some notable quotes from the feedback. On feedback regarding what worked well in the interaction, some people said:

> "Everything seems to function properly especially when the agent replies to me"
>
> "It was very straightforward and quick to get answers. "
>
> "The instructions were spot on"
>
> "I feel the instructions were very easy to follow."
>
> "The step by step instructions"

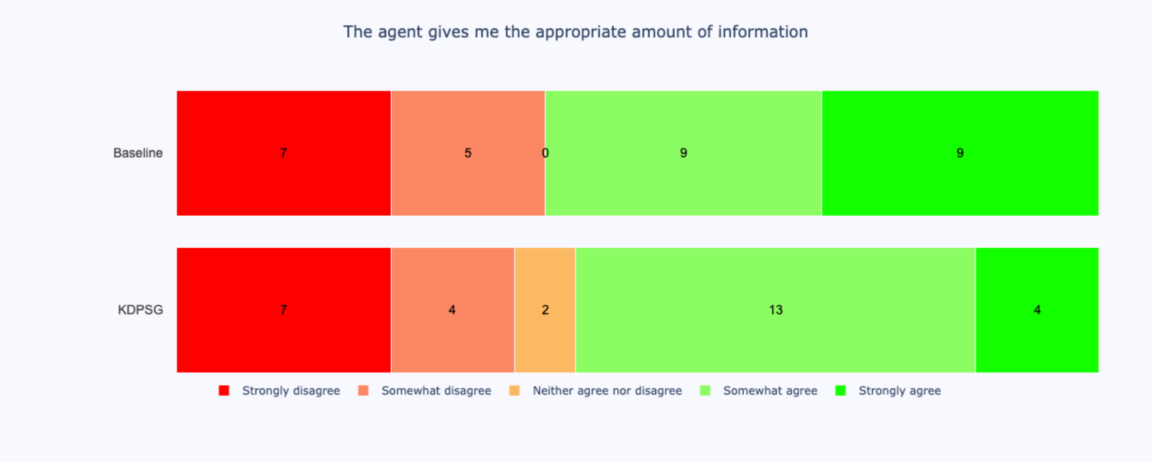


Figure 7-11: Distribution of whether an agent gave an appropriate amount of information or not. We see that most people were neutral.

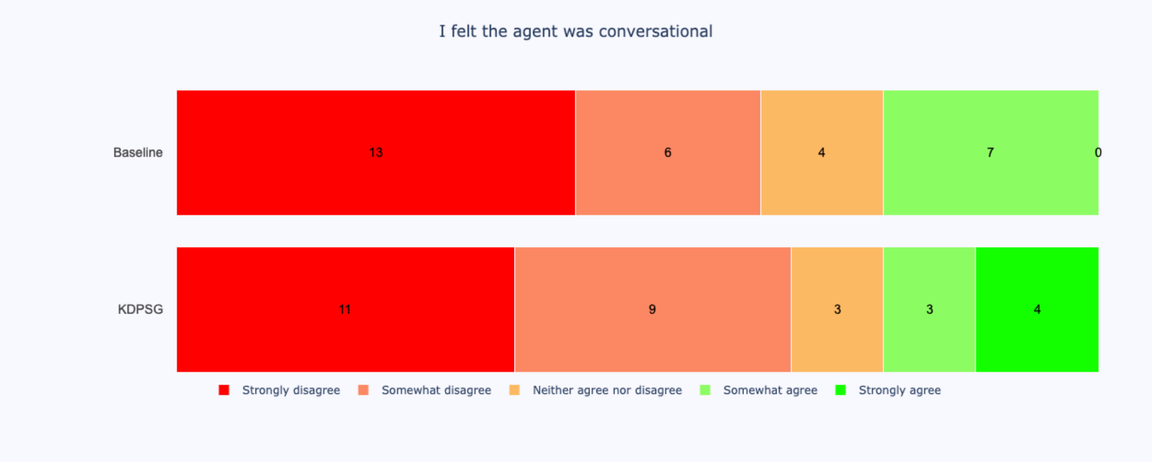


Figure 7-12: Distribution of whether a participant found the agent conversational or not. Most people did not think the agent was conversational.

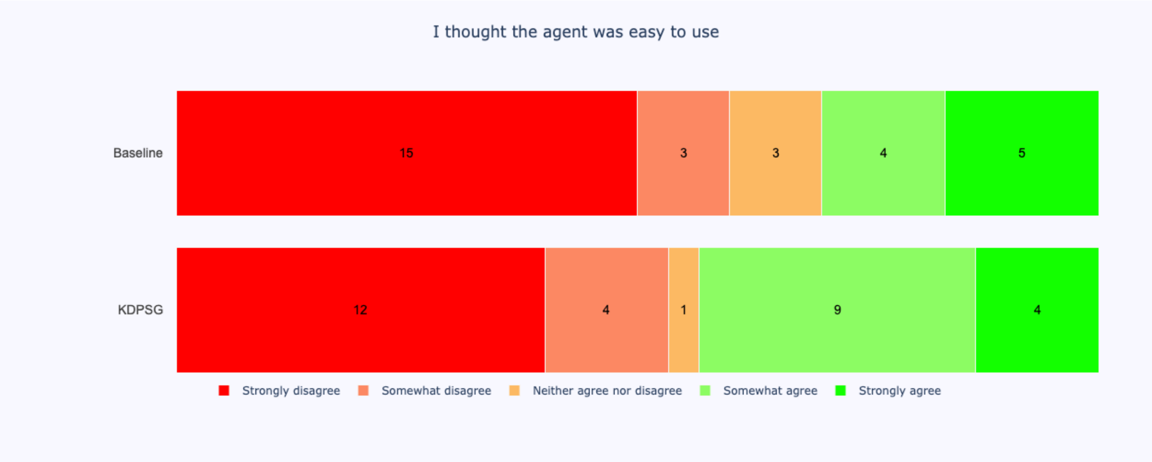


Figure 7-13: Distribution of whether an agent was easy to use or not. Most people did not find the agent easy to use.

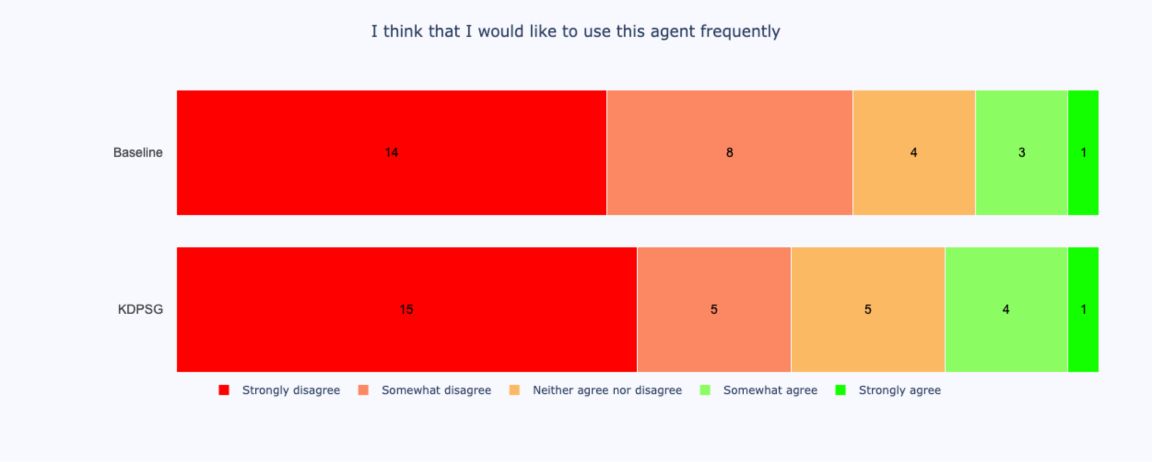


Figure 7-14: Distribution of whether a participant would use the agent frequently or not. Most people would not use the agent frequently.

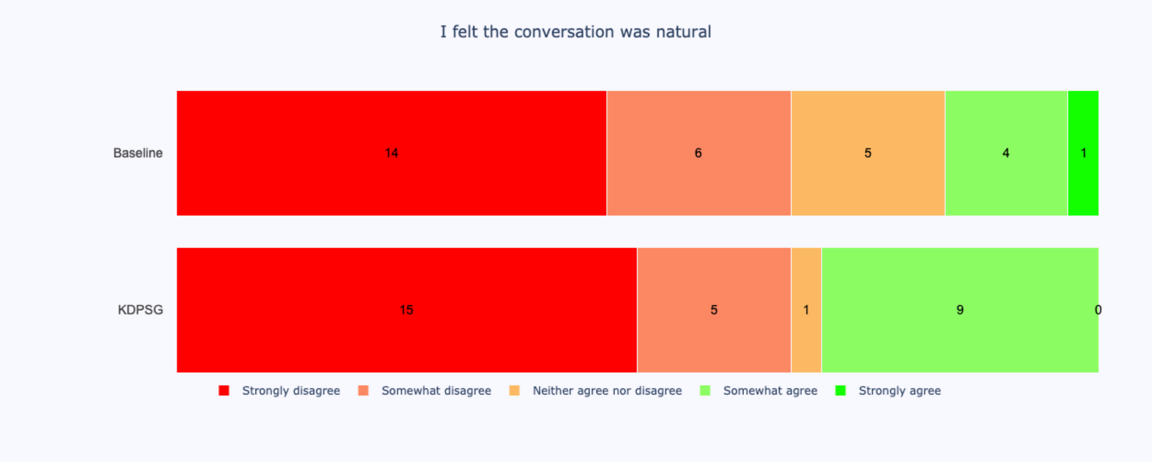


Figure 7-15: Distribution of whether the conversation felt natural or not. Most people thought the conversation was not natural.

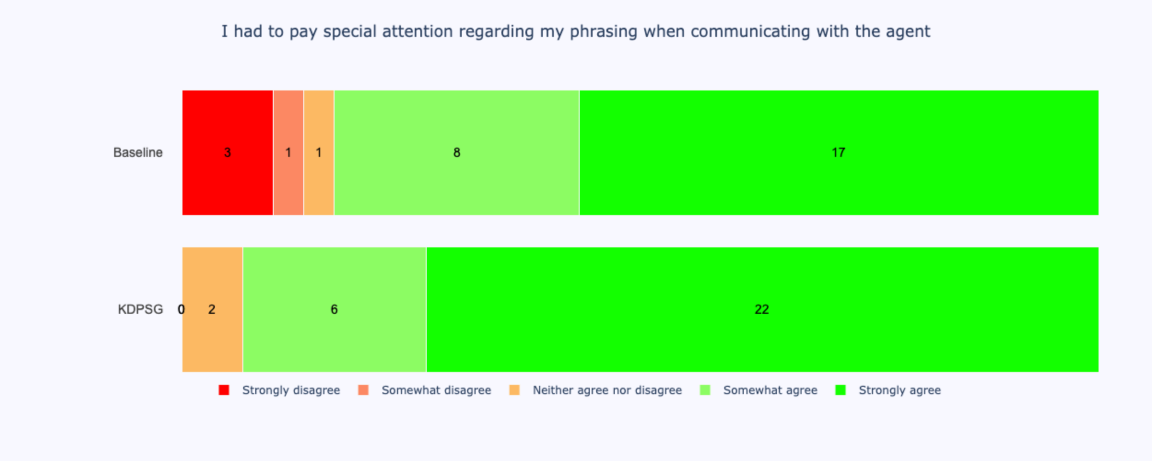


Figure 7-16: Distribution of whether a participant had to use special phrasing or not. Most people had to use special phrasing to communicate effectively.

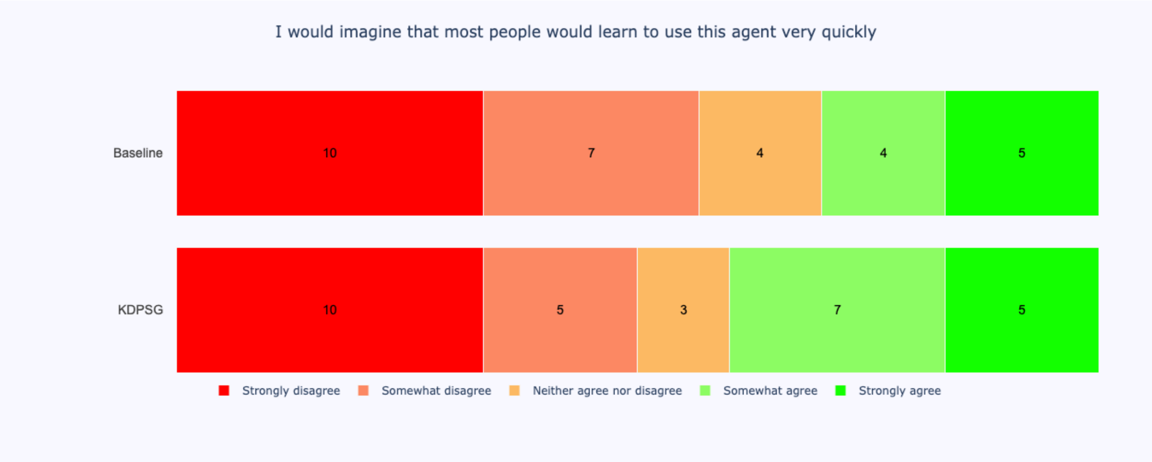


Figure 7-17: Distribution of whether participants would learn to use the agents quickly or not. Most people would not learn to use the agent quickly.

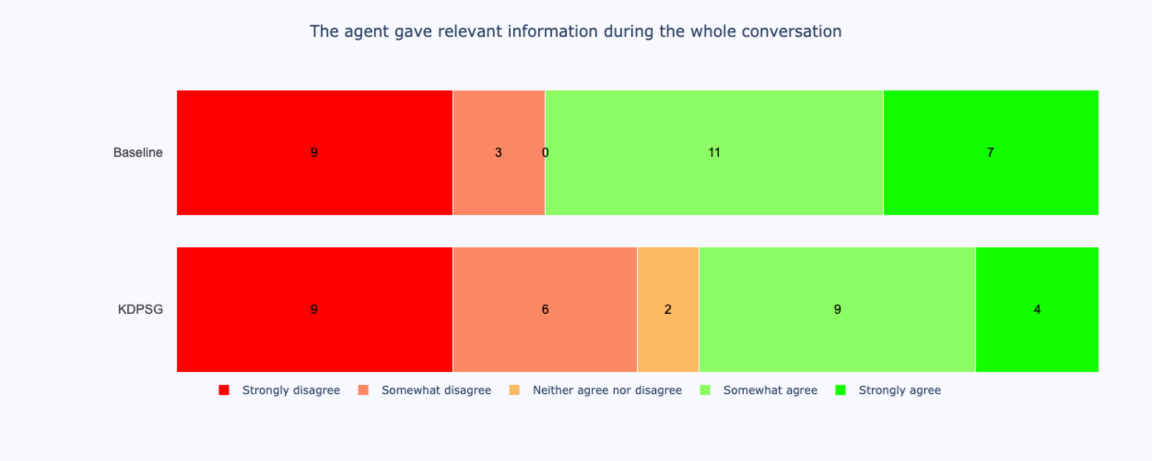


Figure 7-18: Distribution of whether an agent gave relevant information or not. Most people were neutral on the agent giving relevant information.

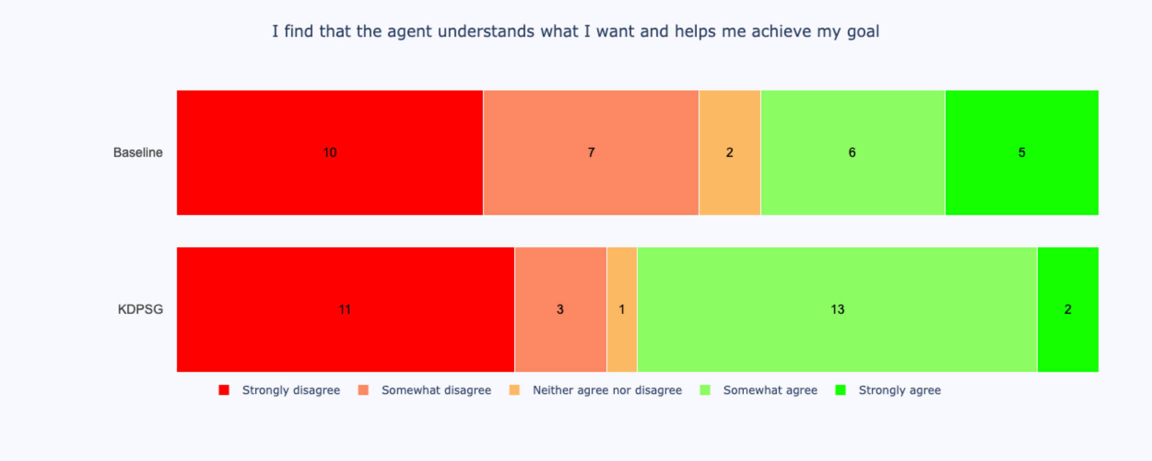


Figure 7-19: Distribution of whether participants thought that the agent understood them and helped them achieve their goals. Most people leaned towards thinking that the agent did not understand them.

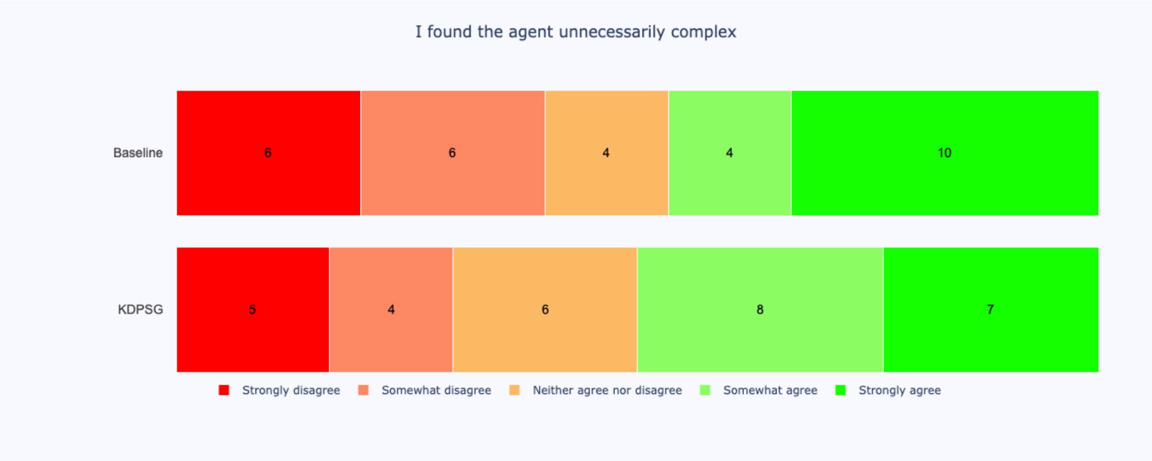


Figure 7-20: Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people were neutral on whether the agent was too complex or not.

In contrast, on things that did not go well in the interaction:

> “Having to rephrase my question multiple times for the bot to understand.”
>
> “Some of my phrases did not work”
>
> “I needed to say "ok" to tell it to move to the next step”
>
> “It wasn’t giving me the what I was asking for ”
>
> “The agent only gave me one step at a time which felt very tedious.”

We now move on to the other block, which consisted of the instruction retrieval system against a ChatGPT powered agent.

Golden Instruction Retrieval vs. GPT

For the second set of conditions (the baseline golden instruction retrieval system vs. the LLM based system), we had 30 participants in our study (N=30), and the same setup of tasks as the prior study. We now go through the results. We find that participants took on average 31.05 minutes to complete the survey. We find that most (N=25 or 83%) of the participants are moderately or more familiar with conversational agents. Only one of the participants was unfamiliar of conversational agents. A view of this distribution can be seen in Figure 7-21. We also saw from the pre-study questions that most people (73%) find conversational agents to be moderately useful or more, as seen in Figure 7-22. Lastly, we see that most participants interact with a conversational agent at least once a day (63%) as seen in Figure 7-23.

Now, in Figure 7-24 we report participant’s familiarity with performing our selection of tasks.

We see that, once more, most tasks, save for composing an email with an embedded image, have been performed “rarely (1-3 times)” or less in the past month. This can confirm that our selection of tasks was good, with 7/8 tasks that people do not perform frequently. From this reported performance, we then sampled 1 task randomly from the set of least performed tasks. The distribution of tasks that a participant was given can be seen in Figure 7-25

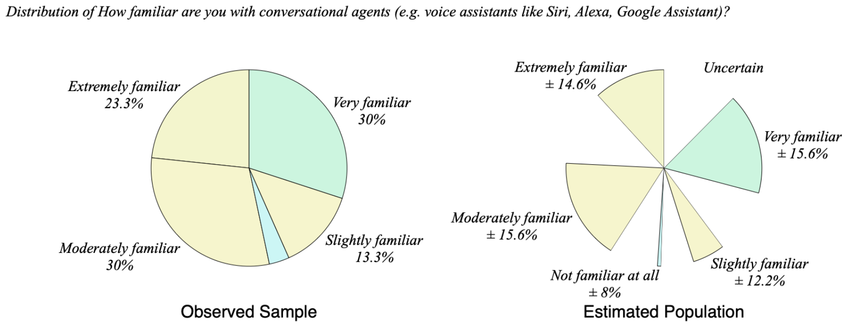


Figure 7-21: Distribution of participants' familiarity with conversational agents.Most people are moderately or more familiar.

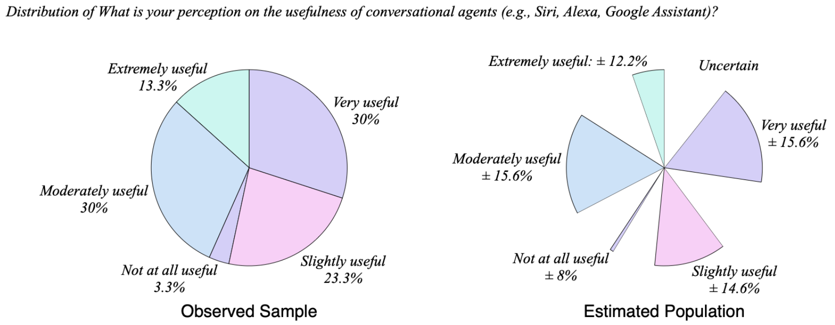


Figure 7-22: Distribution of participant's perception of conversational agents. Most people think they are moderately useful or more.

When we reviewed the feedback of whether the tasks that were given were appropriate or not for a conversational agent, the feedback is generally positive and suggests that the selected task is appropriate for a conversational agent. Many users found the tasks to be simple and easy to follow (24/30), and appreciated the convenience of having an agent guide them through the process instead of having to search for tutorials. Some users expressed doubts about the agent's ability to understand certain questions or perform the task effectively, but overall the feedback was positive.

Now we move on to the more important results from the study in Figures 7-26 and 7-27. From here, we can see that a significant portion of participants were able to complete the task that they were assigned, and that they were able to follow the instructions that an agent gave them. Namely, 25/30 people were able to do what the agent guided them on (goal completion) with ChatGPT, and 19/30 were able to do what the baseline agent told them to do. In addition to this, 27/30

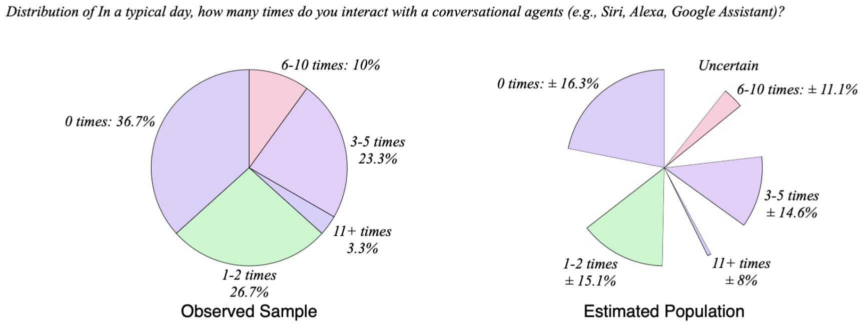


Figure 7-23: Distribution of people's daily interactions with conversational agents. 1/3 of people do not interact with them, and most people interact at least once with them.

participants were able to complete their goal with the ChatGPT agent and 21/30 with the baseline. Both of these results present a significant difference ($p<0.05$) against an estimated success rate of 50% using a binomial test. We note that there is no significant difference between the agents in either case ($p=0.55$ with a Fisher exact test for task completion and $p=0.1$ for goal completion, $p=0.056$ $p=0.07$ respectively for a for χ^2 test), indicating that the quality of the guidance was similar.

Now we look at some of the failures and the successes of the interactions. Overall, with our baseline agent, we see the same behavior as in the other block: if the intent matching and slot filling fail (7/9, 1/9 failed interactions respectively), people cannot receive guidance or the procedure search is completely thrown off to the point that people are not capable of completing the procedure. Vice versa, if it does match, users are able to receive appropriate guidance. In the case of the ChatGPT-based agent, we saw that the failures were because of outdated information (1/3) (in iOS based devices, it gave directions about older applications/settings), and from extremely general phrasing of how to do certain actions (2/3). There is one particular error in the GPT-based agent that is interesting. One participant said:

> *"The agent needs more of an idea of how various email services allow you to set up a signature. I went to where it told me, and I ended up in my Gmail account settings. Nothing to do with the actual email settings."*

Now, we look into the conversation that the user had with the agent that failed. That

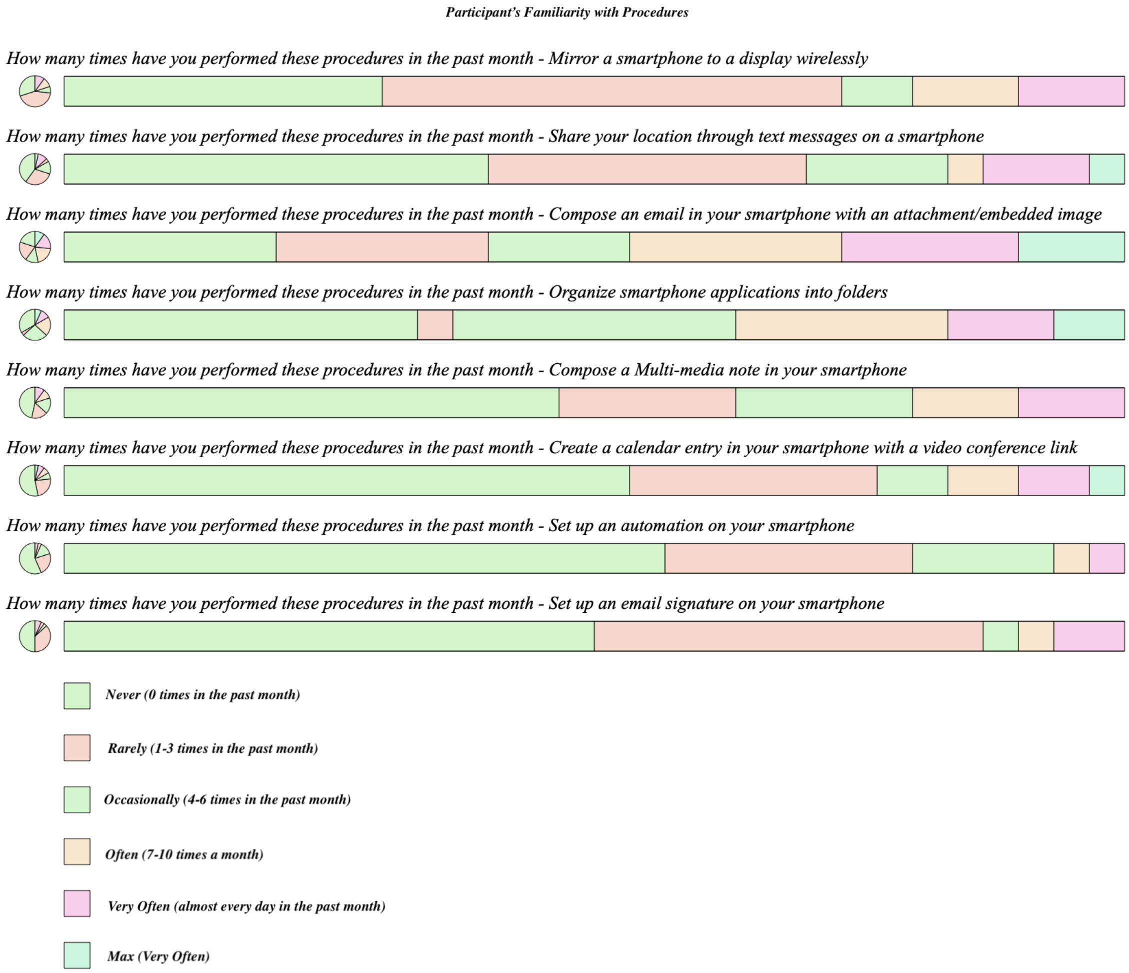


Figure 7-24: Distribution of participants reported monthly performance of tasks. We see that most tasks have been done rarely, except for organizing folders and sending emails with images.

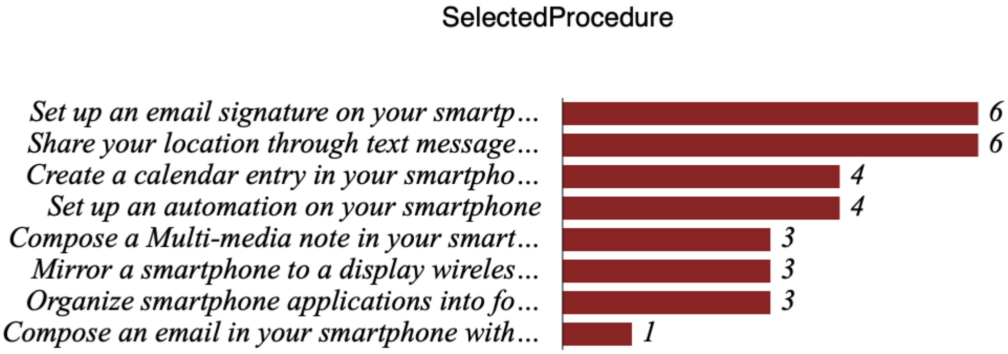


Figure 7-25: Distribution of the procedures that were assigned to participants. Once more, the selection process follows an inverse of the reported performance, with composing an email and organizing into folders being some of the least performed.

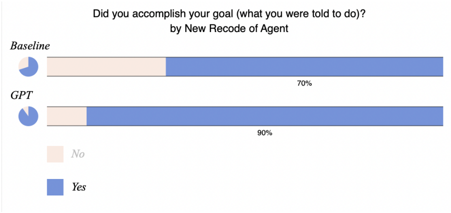


Figure 7-26: Distribution of whether a participant was able to accomplish the goal they were given or not. We see that most interactions (70%+) were able to accomplish their goals with the given agents. The GPT based agent was able to guide more interactions successfully.

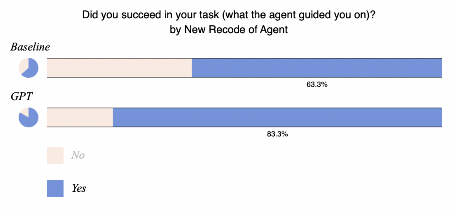
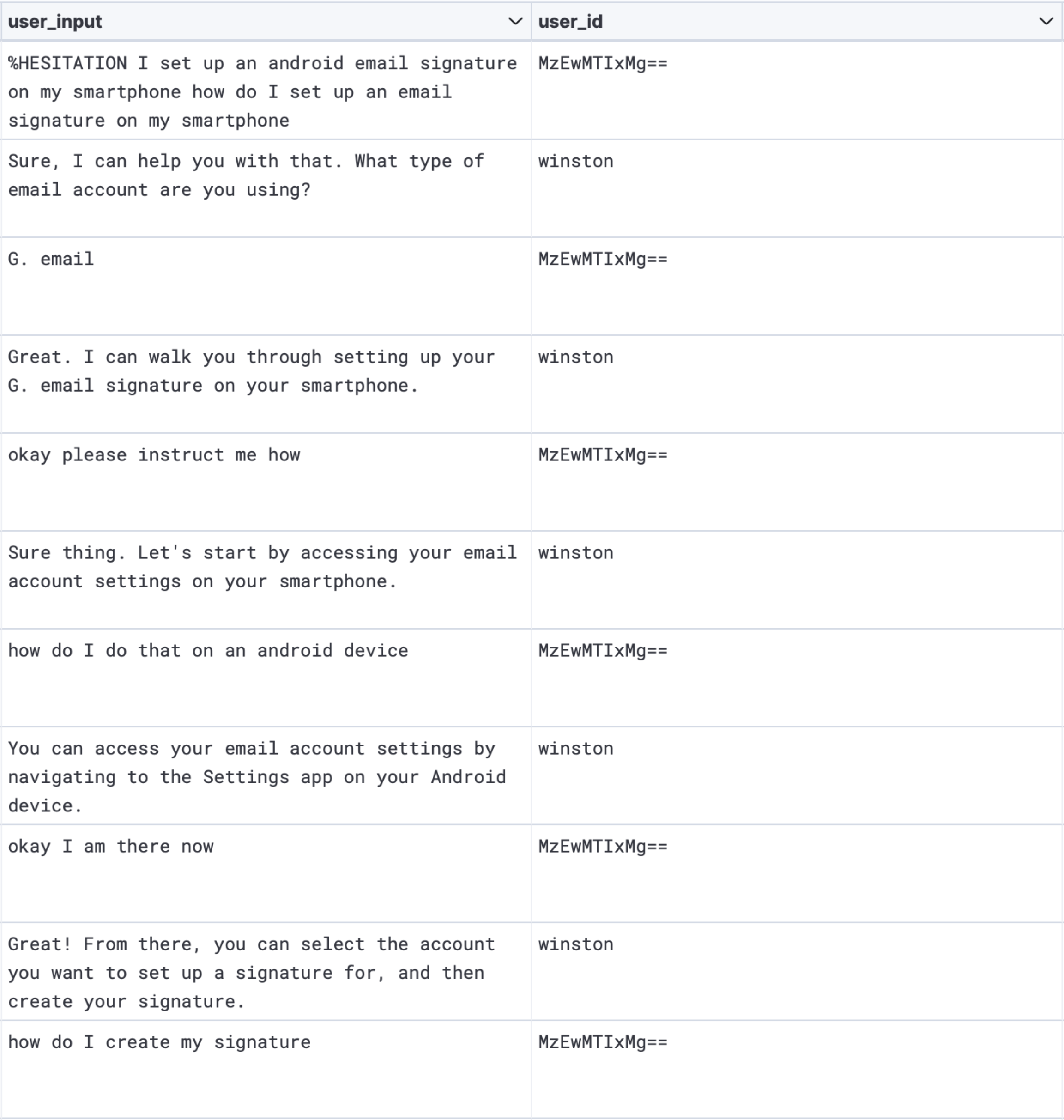


Figure 7-27: Distribution of whether a participant was able to accomplish the instructions that the agent gives. We see that most people were able to follow the instructions that the agents gave. Interactions with the GPT based agent were followed more successfully.

conversation can be seen in Figure 7-28. We can see that this is a combination of a hallucination along with very general instructions on how to set up a signature on an email. Similarly, another user that wanted an automation on their device for a reminder tried to get directions on how to do it, but eventually the agent mixed in two procedures (how to make a shortcut in iOS with how to make a reminder). The interaction can be seen in Figure 7-29.

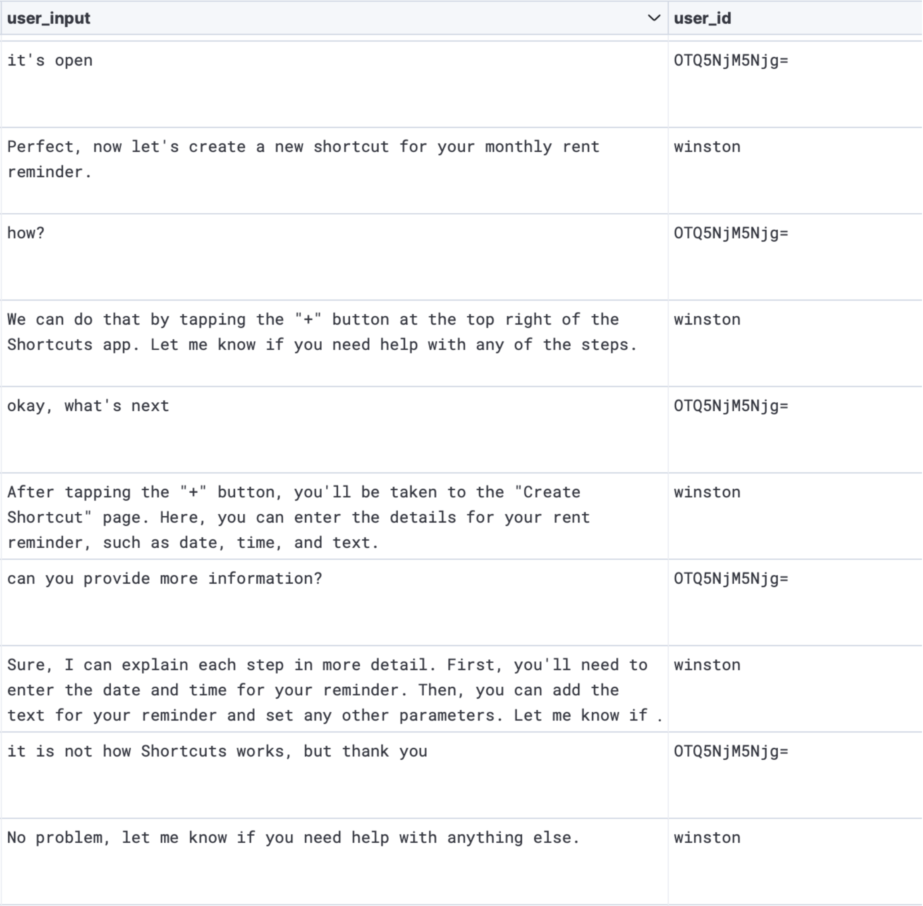
We see that the GPT-based agent knew a broad way of how to make a shortcut, but it did was not able to put together how to do one that sends reminders appropriately. Instead, it gives general instructions on making a reminder. Lastly, we note that the ChatGPT-based agent sometimes does not respond adequately to conversational patterns that we use. What this means is that sometimes the ChatGPT based agent will engage in simple back and forth conversation without moving the conversation forward in the direction that the person wants. In the case seen in Figure 7-30, we see that there is a back and forth of the agent trying to get some details but always informing the person that it's ready to start, even though the person says to start a couple of times. This could be seen as the agent misunderstanding the person's intent to begin receiving guidance on the procedure at hand, and simply continuing chit-chat with a person.

Now, after analyzing the failures and successes, we look at the usability. In all the questions except whether special attention had to be paid to phrasing and the agent was unnecessarily complex, there was a significant difference between the baseline agent and the LLM-based agent ($p>0.05$ with a Wilcoxon signed-rank test). We find that the baseline system performs very similar to the other experimental setup. People disagreed that the baseline was conversational, and were neutral on whether GPT was conversational (Figure 7-32. and there was a significant difference between the ChatGPT-based agent and our baseline ($p=0.001$ with a Wilcoxon signed rank test). Additionally, participants felt that the conversation from the Baseline was not natural, and that conversation from GPT leaned towards somewhat conversational (Figure 7-35, and there was a significant difference between the ChatGPT-based agent and our baseline ($p=0.0001$ in a Wilcoxon signed rank test). Interestingly, we see that



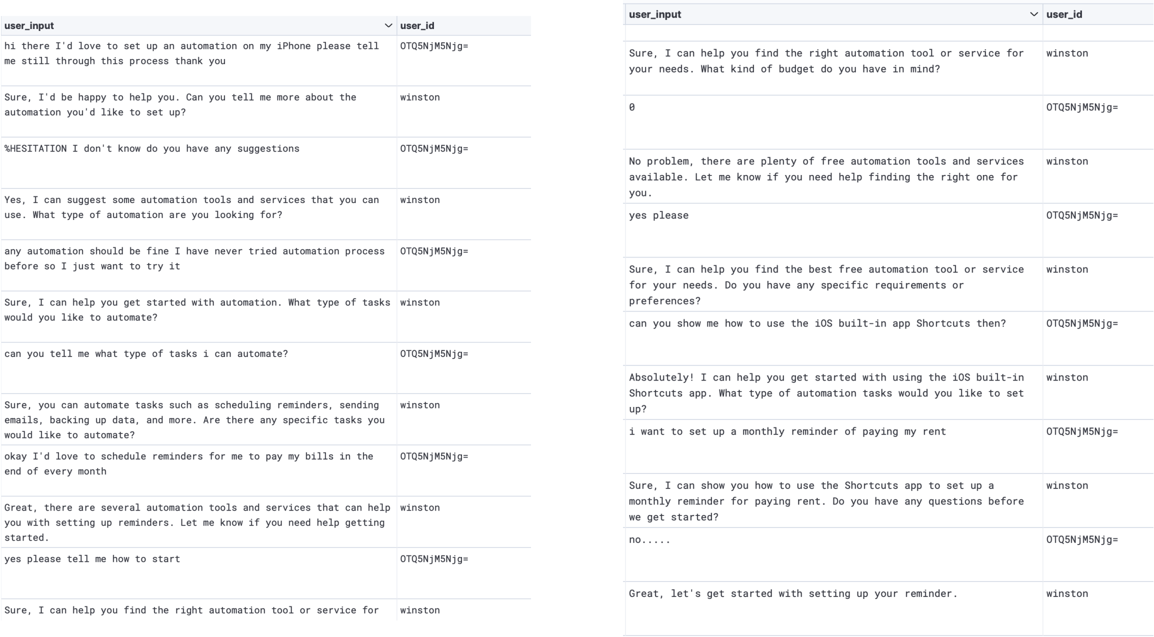
user_input	user_id
%HESITATION I set up an android email signature on my smartphone how do I set up an email signature on my smartphone	MzEwMTIxMg==
Sure, I can help you with that. What type of email account are you using?	winston
G. email	MzEwMTIxMg==
Great. I can walk you through setting up your G. email signature on your smartphone.	winston
okay please instruct me how	MzEwMTIxMg==
Sure thing. Let's start by accessing your email account settings on your smartphone.	winston
how do I do that on an android device	MzEwMTIxMg==
You can access your email account settings by navigating to the Settings app on your Android device.	winston
okay I am there now	MzEwMTIxMg==
Great! From there, you can select the account you want to set up a signature for, and then create your signature.	winston
how do I create my signature	MzEwMTIxMg==

Figure 7-28: Failed GPT-based agent conversation. We can see the GPT-based agent making up very broad instructions on setting a signature in “G. email”



user_input	user_id
it's open	OTQ5NjM5Njg=
Perfect, now let's create a new shortcut for your monthly rent reminder.	winston
how?	OTQ5NjM5Njg=
We can do that by tapping the "+" button at the top right of the Shortcuts app. Let me know if you need help with any of the steps.	winston
okay, what's next	OTQ5NjM5Njg=
After tapping the "+" button, you'll be taken to the "Create Shortcut" page. Here, you can enter the details for your rent reminder, such as date, time, and text.	winston
can you provide more information?	OTQ5NjM5Njg=
Sure, I can explain each step in more detail. First, you'll need to enter the date and time for your reminder. Then, you can add the text for your reminder and set any other parameters. Let me know if .	winston
it is not how Shortcuts works, but thank you	OTQ5NjM5Njg=
No problem, let me know if you need help with anything else.	winston

Figure 7-29: Failed GPT-based agent conversation. We can see once more that the GPT agent tries to make very broad instructions on how to make a reminder Shortcut. Eventually, the agent mixes in the information of how to make a general reminder, but it does not mix it well within how to make the automation.



user_input	user_id
hi there I'd love to set up an automation on my iPhone please tell me still through this process thank you	OTQ5NjM5Njg=
Sure, I'd be happy to help you. Can you tell me more about the automation you'd like to set up?	winston
%HESITATION I don't know do you have any suggestions	OTQ5NjM5Njg=
Yes, I can suggest some automation tools and services that you can use. What type of automation are you looking for?	winston
any automation should be fine I have never tried automation process before so I just want to try it	OTQ5NjM5Njg=
Sure, I can help you get started with automation. What type of tasks would you like to automate?	winston
can you tell me what type of tasks i can automate?	OTQ5NjM5Njg=
Sure, you can automate tasks such as scheduling reminders, sending emails, backing up data, and more. Are there any specific tasks you would like to automate?	winston
okay I'd love to schedule reminders for me to pay my bills in the end of every month	OTQ5NjM5Njg=
Great, there are several automation tools and services that can help you with setting up reminders. Let me know if you need help getting started.	winston
yes please tell me how to start	OTQ5NjM5Njg=
Sure, I can help you find the right automation tool or service for	winston

user_input	user_id
Sure, I can help you find the right automation tool or service for your needs. What kind of budget do you have in mind?	winston
0	OTQ5NjM5Njg=
No problem, there are plenty of free automation tools and services available. Let me know if you need help finding the right one for you.	winston
yes please	OTQ5NjM5Njg=
Sure, I can help you find the best free automation tool or service for your needs. Do you have any specific requirements or preferences?	winston
can you show me how to use the iOS built-in app Shortcuts then?	OTQ5NjM5Njg=
Absolutely! I can help you get started with using the iOS built-in Shortcuts app. What type of automation tasks would you like to set up?	winston
i want to set up a monthly reminder of paying my rent	OTQ5NjM5Njg=
Sure, I can show you how to use the Shortcuts app to set up a monthly reminder for paying rent. Do you have any questions before we get started?	winston
no.....	OTQ5NjM5Njg=
Great, let's get started with setting up your reminder.	winston

Figure 7-30: Failed GPT-based agent conversation. We can see the agent always trying to get more information and not starting the process even though the user indicates this.

participants tended to agree that in both systems one would need to pay attention when addressing the systems (Figure 7-36), and there was no significant difference (p=0.055 in a Wilcoxon signed rank test) between agents in this perception. People also leaned towards disagreeing that the Baseline agent understood and tried to help on what wanted to be done and leaned towards agreeing that the ChatGPT agent understood and tried to help (Figure 7-39), and there was a significant difference between the ChatGPT-based agent and our baseline (p=0.002 in a Wilcoxon signed rank test). Another interesting observation is that participants would somewhat not want to use either agent frequently, and that they leaned towards the agent not being complex (Figures 7-34 and 7-40 respectively), with a significant difference (p=0.01) and no significant difference (p=0.25 on Wilcoxon signed rank test) respectively). Now, we look at the final set of usability metrics, which are that participants neutral that the baseline gave relevant information, and participants leaned towards somewhat agreeing that the ChatGPT based agent gave relevant information (Figure 7-38) and there was a significant difference between both agents (p=0.04 for paired Wilcoxon signed rank test). Most participants disagreed that the baseline was easy to use, and

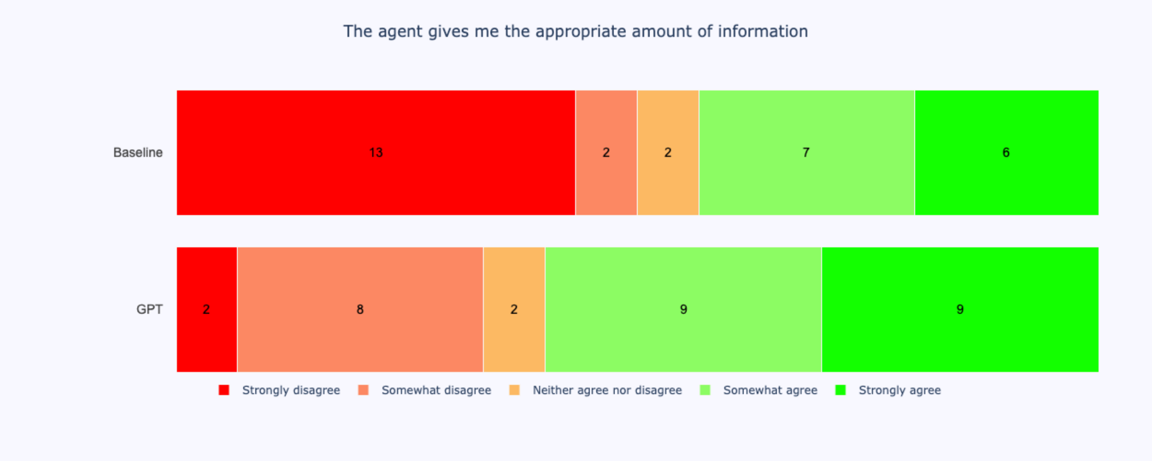


Figure 7-31: Distribution of whether an agent gave an appropriate amount of information or not. People thought the GPT based agent gave more appropriate amounts of information and leaned towards disagreeing on the baseline

leaned towards agreeing that the ChatGPT agent was easy to use, with a significant difference as seen in Figure 7-33 between the agents (p=0.002 for a Wilcoxon signed rank test). Finally, participants leaned towards disagreeing that people would learn to use the agent quickly, and participants leaned towards agreeing this for the ChatGPT-based agent (Figure 7-37). There was a significant difference (p=0.004 for a Wilcoxon signed rank test) between the two agents.

From these results we get the following. One of the major differences between the baseline agent, and the GPT based agent, is the flexibility with which the GPT based agent can respond to user's input. The GPT agent can take spelling errors as input and produce effective outputs, and people can ask for the general version of a procedure ("how to do [X] in my smartphone" instead of "how to do [X] in my [iPhone/Android]), and later on fine tune details through the conversation. This flexibility, which our intent-based system does not have, we believe is what gives the ChatGPT-based agent improved usability metrics in all the categories that we evaluated. Users were usually able to, with the first query or two, receive guidance on the ChatGPT-based agent, whereas if intent matching failed on the baseline, they would need to rephrase a few more times until it picked up.

Finally, we look at some quotes regarding what went well during the interaction:

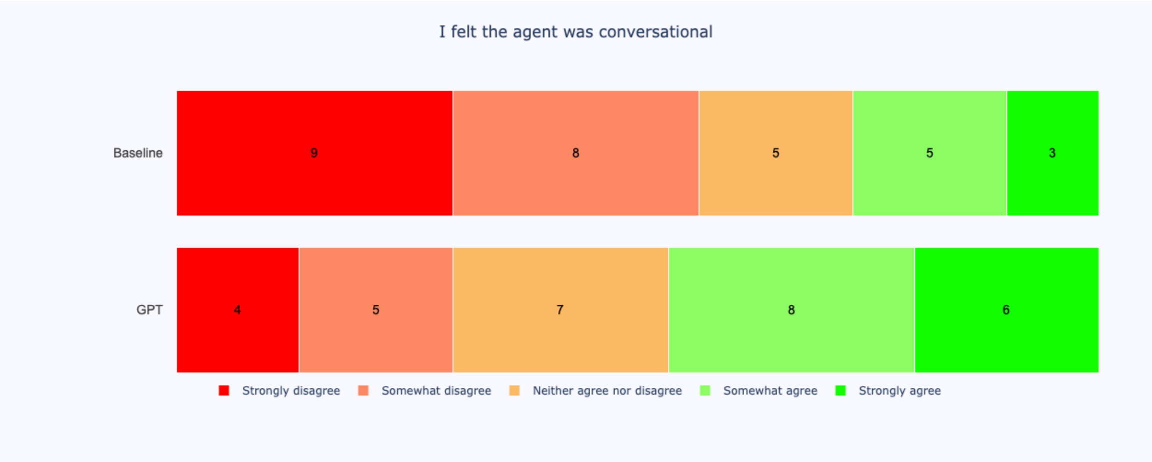


Figure 7-32: Distribution of whether a participant found the agent conversational or not. People found the GPT agent conversational in contrast to the baseline agent.

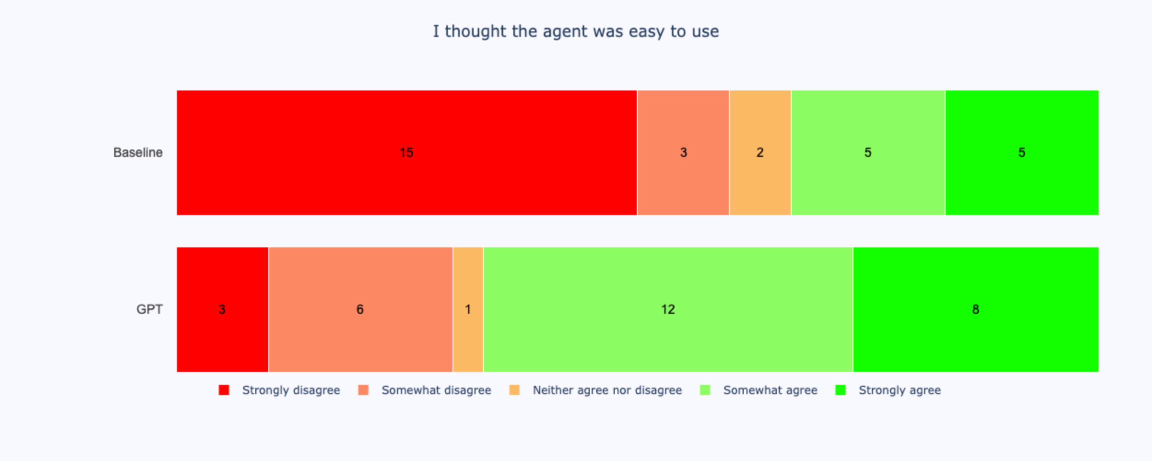


Figure 7-33: Distribution of whether an agent was easy to use or not. People leaned towards agreeing on the GPT based agent, in contrast to disagreeing on the baseline.

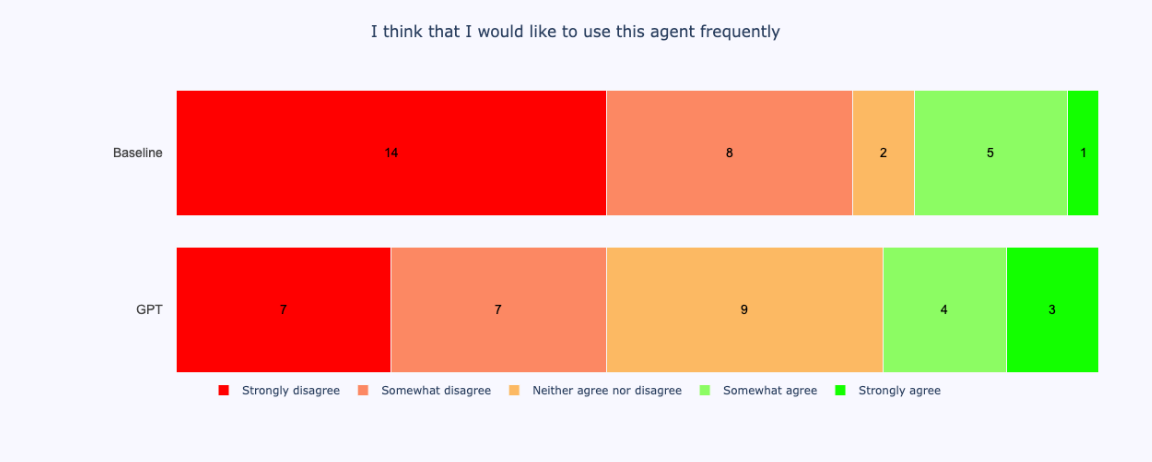


Figure 7-34: Distribution of whether a participant would use the agent frequently or not. In both cases, people leaned towards not utilizing the agents frequently.

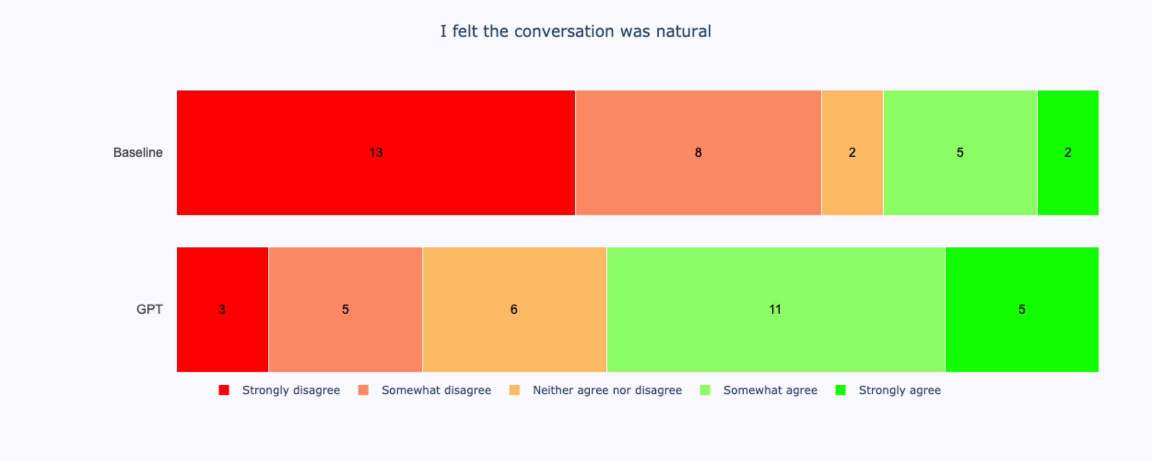


Figure 7-35: Distribution of whether the conversation felt natural or not. In the case of the GPT based agent, people leaned towards agreeing it was natural, in contrast to disagreeing with the baseline.

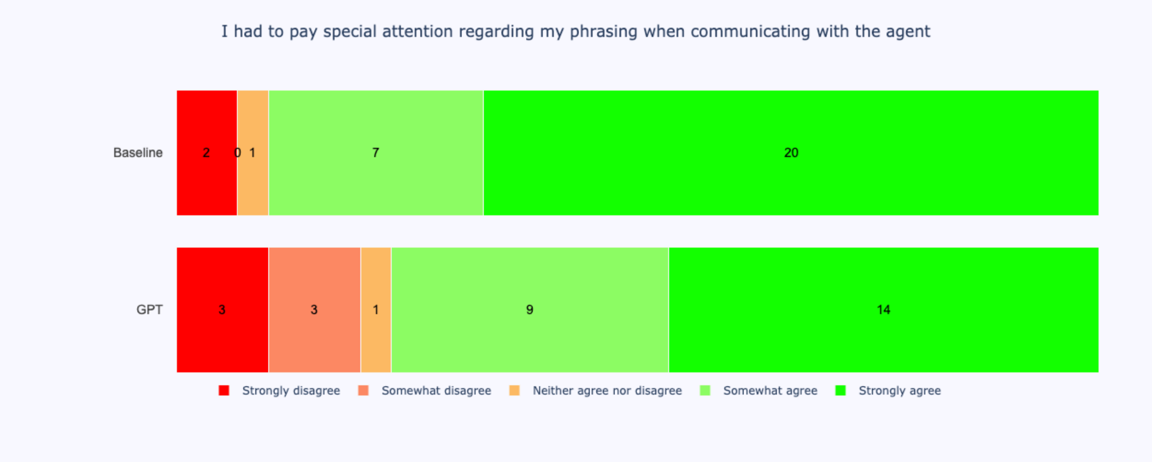


Figure 7-36: Distribution of whether a participant had to use special phrasing or not. Interestingly, people felt that they needed special phrasing for both agents.

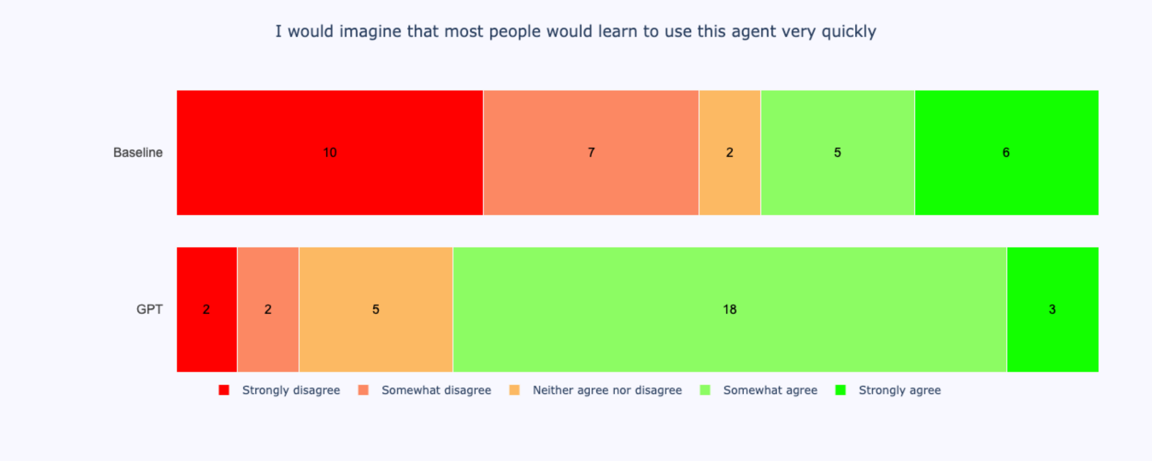


Figure 7-37: Distribution of whether participants would learn to use the agent quickly or not. People leaned towards disagreeing with the baseline agent, and agreeing with the GPT based agent.

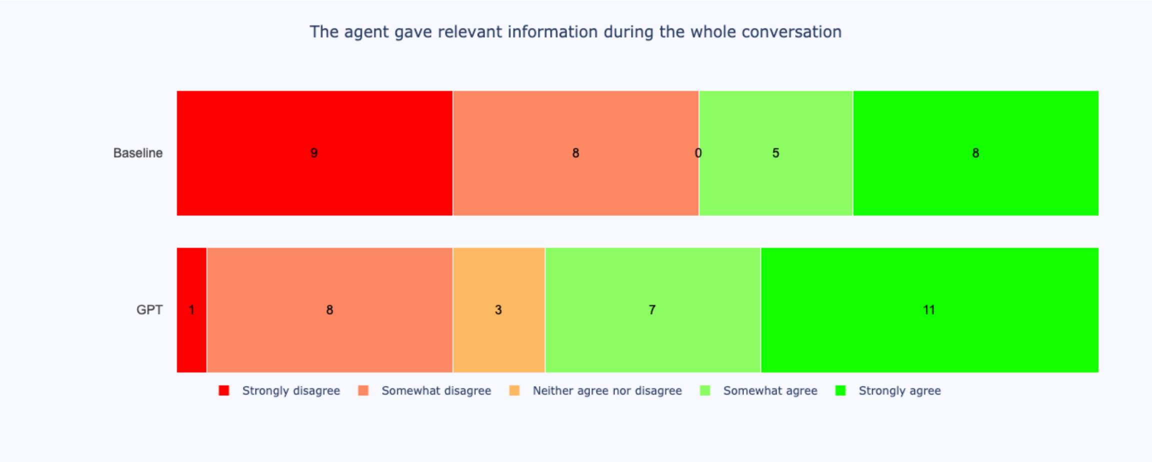


Figure 7-38: Distribution of whether an agent gave relevant information or not. People were neutral to positive that both agents gave relevant information.

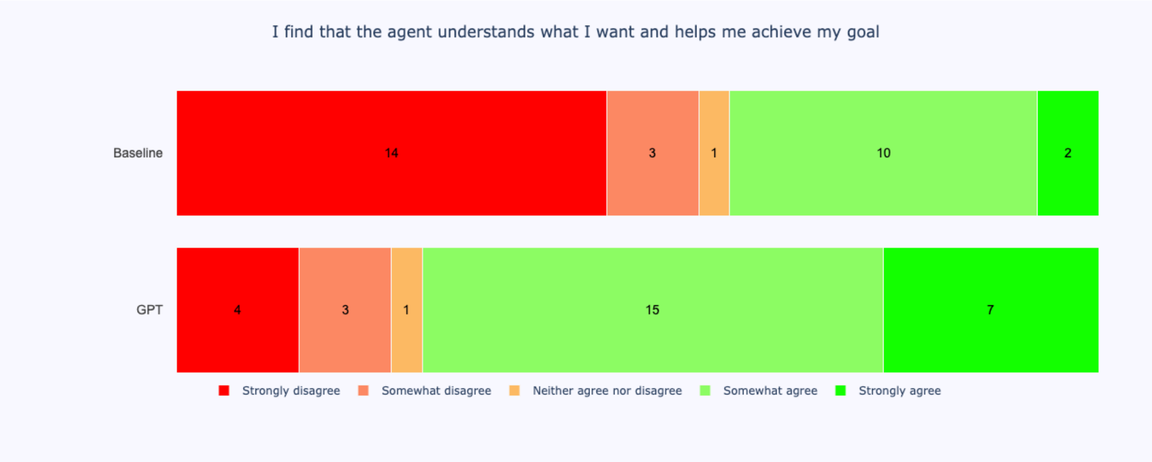


Figure 7-39: Distribution of whether participants thought that the agent understood them and helped them achieve their goals.People thought the GPT based agent understood them more than the baseline.

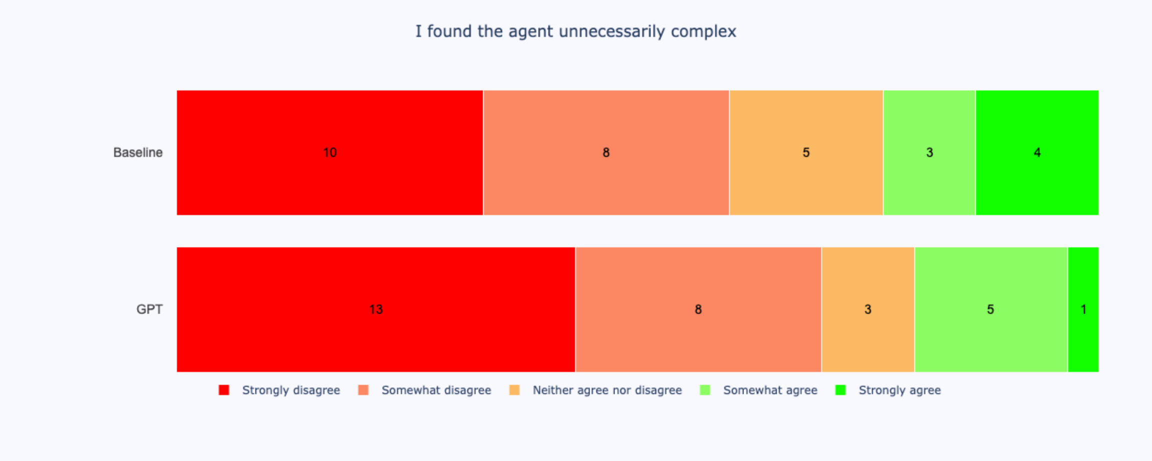


Figure 7-40: Distribution of whether a participant thought that an agent was unnecessarily complex or not. People disagreed that both agents were complex.

> "The agent worked really well and was very clear in what was said to be done"
>
> "All instructions were exact to what was happening on my phone"
>
> "All the steps provided were clear and precise, worked well"
>
> "The agent understood everything i asked/said"

In contrast, things that did not go well while doing the interaction:

> "needed to prompt to get next step"
>
> "I need to search a bit for specific tools, images would be nice"
>
> "Had to push the AI to go to the next step"
>
> "I had to ask the right question"
>
> "The buttons I was told to use they don't exist on either phone the three dots do not either not wear I was instructed to anyways"

7.2.5 Overall Findings

We now list some of the findings from this study:

- Our results indicate that all three of the conversational agents tested can be utilized for procedural guidance on the tasks that were given [H1].

- Our results indicate that our generated steps are comparable in effectiveness for procedural guidance as gold-standard instructions [H2].
- Our results indicate that our intent-based system is not very usable and is not natural nor conversational [H3].
- Our results suggest that usability problems our baseline and knowledge-driven step generation system mostly focus on problems related to intent matching and slot filling, as users had to rephrase multiple times for the system to understand them and use very specific vocabulary to get adequate guidance. [H4]
- Our results suggest that in contrast to the baseline, the LLM based agent's flexibility to parse the user's intent along and respond adequately, influence heavily on people's perception of the usability of the agent. [H4]
- Our results show that the LLM based model is not perfect, notably it sometimes conveys information that is outdated, or does not follow established conversational patterns for driving interactions. [H4]

7.3 Evaluation 2: Error Recovery

7.3.1 Overview

Now that given that we have shown that it is indeed possible to receive guidance from conversational agents, including our knowledge driven agent, we set out to see if people can utilize the same agents, with some error recovery functionality, to fix errors that might arise in a procedure. Our results show that all three agents are indeed capable of guiding people, however similar to our first study, the intent-based agent needs work on the understanding capabilities.

7.3.2 Research Questions

The explicit questions that we set out to solve with this experiment are the following:

1. H5: Can a conversational agent with patterns guide people through procedure error recovery?
2. H6: Can generated instructions be as effective as gold-standard ones in error recovery?
3. H7: Is this error-recovery capable agent natural and conversational?
4. H8: Can we get design feedback on our error-recovery capable agent?

7.3.3 Study Design

In our second study, we utilized almost the same format as the first evaluation in Section 7.2.3. Since we wanted to evaluate the performance of the agents when exceptions were handled, we came up with the strategy that within one of the given tasks, we would tell the participant to stop at a given step, and listen to an error that happened. Then the participants were told to utilize the agent that they were utilizing to address the error and continue the procedure. We selected the following procedures, and a corresponding error within the procedure.

1. Share your location through text messages on a smartphone
    - Users were told to change the duration of the location share, as they had picked a duration that was incorrect
2. Compose an email in your smartphone with an attachment/embedded image
    - Users were told to replace the image that was embedded as it was the wrong image
3. Organize smartphone applications into folders
    - Users were told that they had put the wrong app in a folder
4. Create a calendar entry in your smartphone with a video conference link
    - Users were told that instead of the links that the calendars provide, that they wanted to utilize a Zoom link

We aimed to keep our study as varied as possible (with regards to devices), so once more we did not specify a particular type of smartphone for the tasks. We note that in one of the tasks, we did pick a specific conference link provider (Zoom). Once more, we use gold-standard procedures for the errors and procedures in our backend. In particular, we made sure to have the most up-to-date version of these procedures for both iPhone and Android users. Again, we opted to perform this study with the Prolific[3] platform and use URL query parameters to generate links that participants can access different agents with. We also use the same within-subjects design [13] to maximize the population of the study.

In the process, we first asked users to rate their frequency of performing the tasks that were listed above on a Likert scale of 1-5, with 1 indicating they had not performed the task in the past month and 5 indicating they performed the task almost daily. From the tasks that users had performed the least, we randomly selected one and had the user ask for guidance on how to perform the task using a voice agent. We had the user perform the task twice and provide feedback each time, with the guidance of a different agent. We conducted the study in blocks of two, with the baseline agent compared against the other agent, which may be the GPT agent or our KDPSG. We note that we indicated that users should try to utilize the microphone as much as possible, but that they were free to utilize the keyboard and write to the agent if there were issues. In contrast to the first study, we indicate that users should pause at a certain point in a procedure:

1. *Share your location through text messages on a smartphone*: After selecting the duration of the location share, come back here and continue on to the next page in the study.

2. *Compose an email in your smartphone with an attachment/embedded image*: After selecting the image, come back here and continue on to the next page in the study.,

3. *Organize smartphone applications into folders*: After making a folder with apps,

---

[3] `https://www.prolific.co/`

come back here and continue on to the next page in the study.,

4. *Create a calendar entry in your smartphone with a video conference link*: When the agent tells you to select the Facetime or Google Meet link, come back here and continue on to the next page in the study.

After the users pause, they are informed of the error that has happened and are told to use the agent to help fix this error. Following this, they fill out a feedback form, and move on to repeat the process with a different agent. We note that most of the study questions are the same as in the first study in Section 7.2.3. The only difference is the inclusion of the following error recovery questions:

- Did you succeed in the mistake fixing task (what the agent guided you on to fix the mistake)?
- Were you able to recover from the problem that was given [ERROR DESCRIPTION] and finish the original task? Describe any issues or things that went well.
- How would you ask a virtual agent for help if you made a mistake following instructions?

We opted to take this approach, rather than giving out a wrong instruction, because we felt that people may lose faith/perceive the agent as not being capable of general guidance.

7.3.4 Study Results

Golden Instruction Retrieval vs. KDPSG

We had 30 participants in our study (N=30). We now go through the results of our survey. We find that participants took on average 50.33 minutes to complete the survey. We find that most (N=27 or 90%) of the participants are moderately or more familiar with conversational agents, and that a minority of participants are slightly familiar with them. None of the users were unfamiliar of conversational agents. A

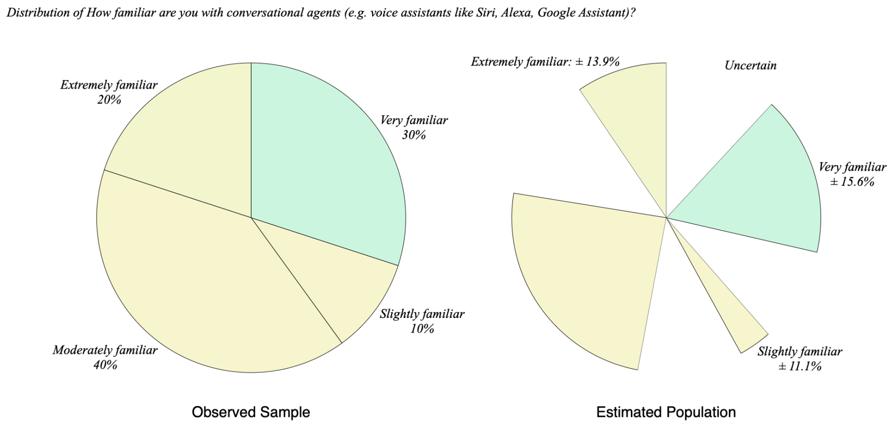


Figure 7-41: User distribution of familiarity with conversational agents. We see most participants were mostly familiar with voice agents.

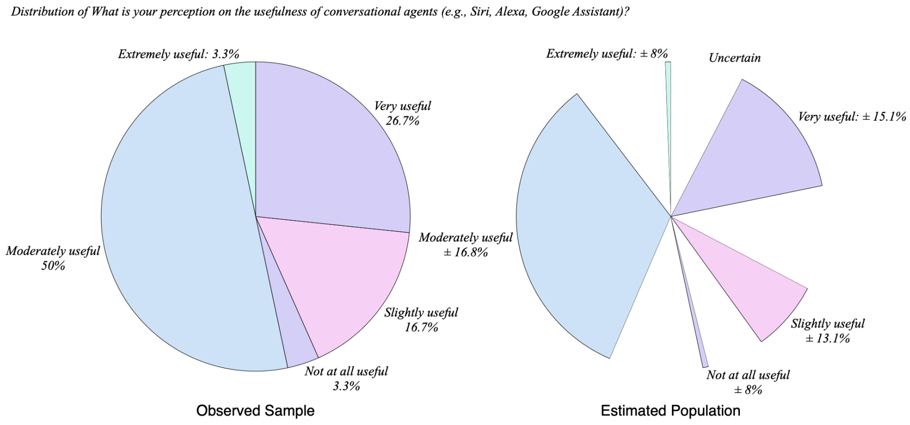


Figure 7-42: Distribution of participant's perception of general usefulness of conversational agents. We see that most people perceive conversational agents as being moderately useful or more.

view of this distribution can be seen in Figure 7-41. We also saw from the pre-study questions that most people (80%) find conversational agents to be moderately useful or more, as seen in Figure 7-42. Lastly, we see that most participants interact with a conversational agent at least once a day (70%) as seen in Figure 7-43.

Now, in Figure 7-44 we report the participant's familiarity with tasks.

We see, similar to the first study, that most tasks, save for composing an email with an embedded image, have been performed close to "rarely (1-3 times)" or less in the past month. We note that this is means that our selection of tasks was decent, with 3/4 tasks that people do not perform frequently. We do note that in this sample

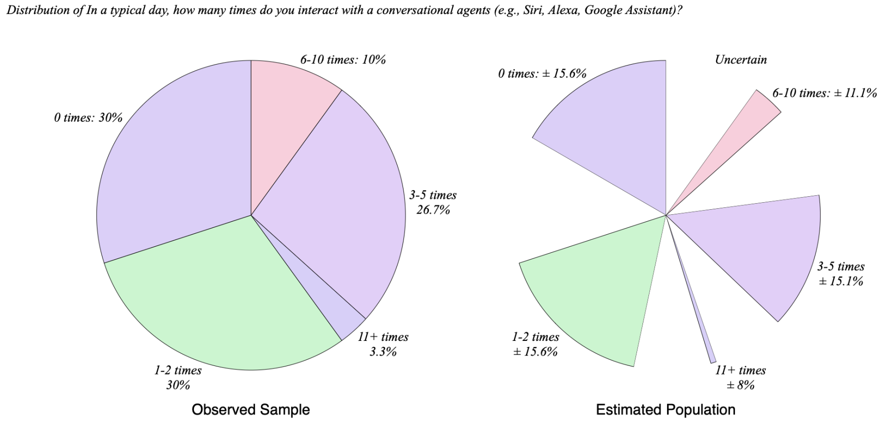


Figure 7-43: Distribution of participants reported interaction quantities with conversational agents. We see that 2/3 of participants use their assistants 1 time or more daily.

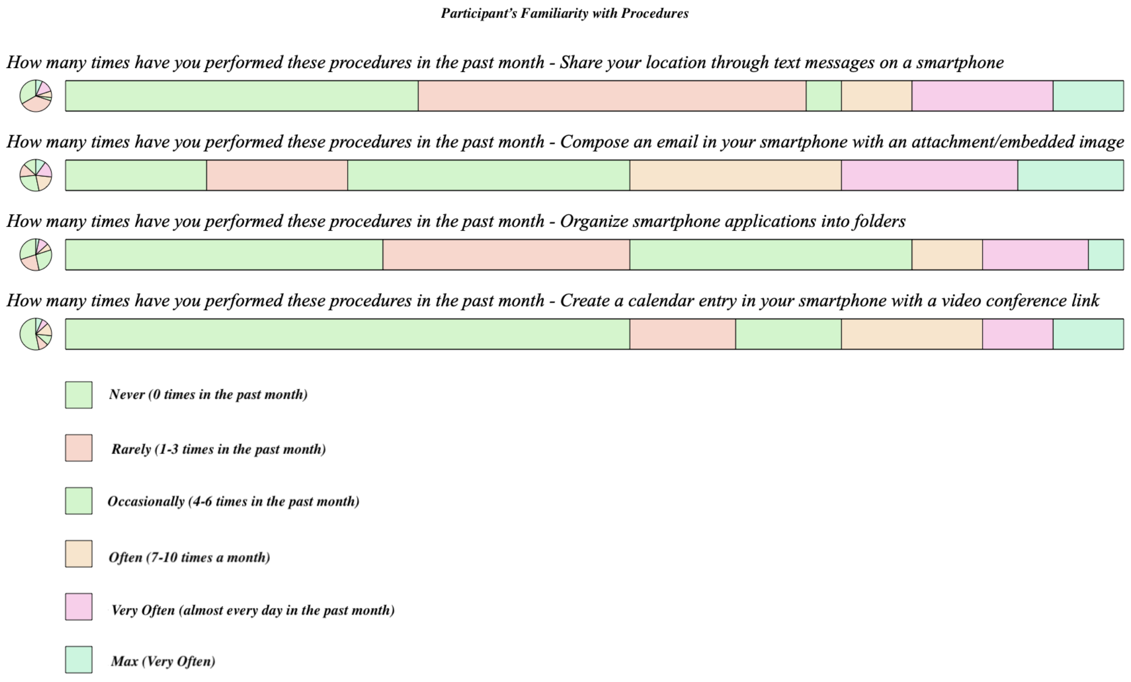


Figure 7-44: Distribution of participants reported monthly performance of tasks. We see once again relatively low frequency of performing all task save sending an email with an image attachment.

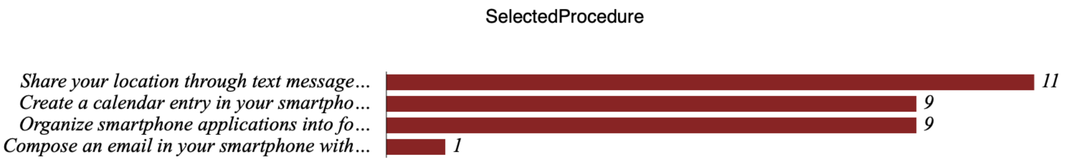


Figure 7-45: Distribution of the procedures that were assigned to participants.

of participants, the averages for the tasks were higher than in the prior study. From this reported performance, we then sampled 1 task randomly from the set of least performed tasks. The distribution of tasks that a participant was given can be seen in Figure 7-45. We note that since users reported the email task as the most popular/familiar, this was the task that was sampled the least (1/30 participants were given this task).

When then moved on to review the feedback of whether the tasks that were given were appropriate or not for a conversational agent. Overall, the feedback suggests that the tasks given to the agents were appropriate and easy enough for them to handle (20/30). However, some users felt that the agents were not always able to understand their queries and did not seem to remember previous interactions, which might be essential for this task. Some found the agents to be helpful for simple tasks, particularly for technical issues (3/30).

Now we move on to the more important results from the study. Some important results can be seen in Figures 7-46, 7-47, 7-48. From here, we can see that a significant portion of participants (26/30 in each group) were able to complete the task that they were assigned, and slightly more (27/30) were able to follow the instructions that an agent gave them. The most important of all results of this study can be seen in Figure 7-48, which is that for both agents, more than half of the participants were able to recover from the error that they were given (36/60, for the baseline 17/30 and for the KDPSG agent 19/30). We note that this number as is in both cases is insignificant, if we perform a binomial test with an estimated 50% success rate ($p>=0.05$), and there is no significant difference between both agents ($p>0.05$ with Fisher's exact test).

Now, there is a significant difference between this amount, and the amount of interactions that were able to complete the original process (p=0.0017, with Fisher exact). We looked into the cases where these agents failed (24/60 interactions) to examine what happened. We found that in most of the interactions that did fail, it was due to some intent matching/slot-filling issue (17/24 or 70% of the interactions that failed). An example (and other interactions were very similar to this) of this can be seen in Figure 7-49. Additionally, there was 1 interaction that failed completely because of the speech-to-text, and 4/17 that failed were because of a transcription error. We note that if we were to remove these interactions that suffered from technical issues, that the overall recovery rates would be 85% (36/42 for both interactions). Individually, for the KDPSG agent, we would have 17/21 (80.9% of interactions), and for the baseline agent we would have 19/21 (90.4% of the interactions) in which the error recovery was successful which are both significantly higher ($p<0.05$) when performing a binomial test with an estimated 50% chance of completing the error recovery. We note that this analysis of errors highlights even more than need for flexible, intent, matching. We saw from the feedback of how people would describe that they made a mistake that many people would simply say a statement without indicating that there was a mistake. This makes it especially hard for the intent recognition system to determine that it should go into a recovery mode. We looked into the open text feedback given from the survey question: "Were you able to recover from the problem that was given ([ERROR]) and finish the original task? Describe any issues or things that went well". There were several instances of negative feedback where users expressed frustration with the conversational agent's inability to understand their requests or respond appropriately (10 interactions reported this). Some users had to restart the conversation multiple times or reword their requests several times before the agent understood them. In some cases, the agent was unable to address the user's issue at all, and some users found the agent unresponsive or unhelpful. Additionally, there were some instances where the agent gave unclear or irrelevant instructions, or where users found the conversation to be confusing or nonsensical.

Now we move on to the usability questions. We see a very similar distribution of usability results as in Evaluation/Study 1. In all the examined metrics, both systems

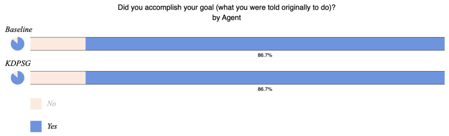


Figure 7-46: Distribution of whether a participant was able to accomplish the goal they were given or not. Most people were able to complete the original task, even if they were not able to recover from an error.

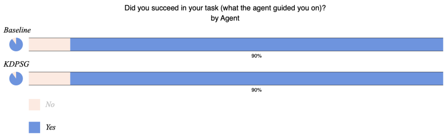


Figure 7-47: Distribution of whether a participant was able to accomplish the instructions that the agent gives. Most people were able to follow the instructions that the agent gave.

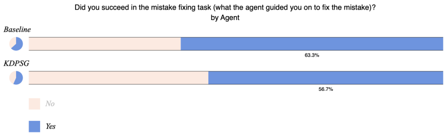
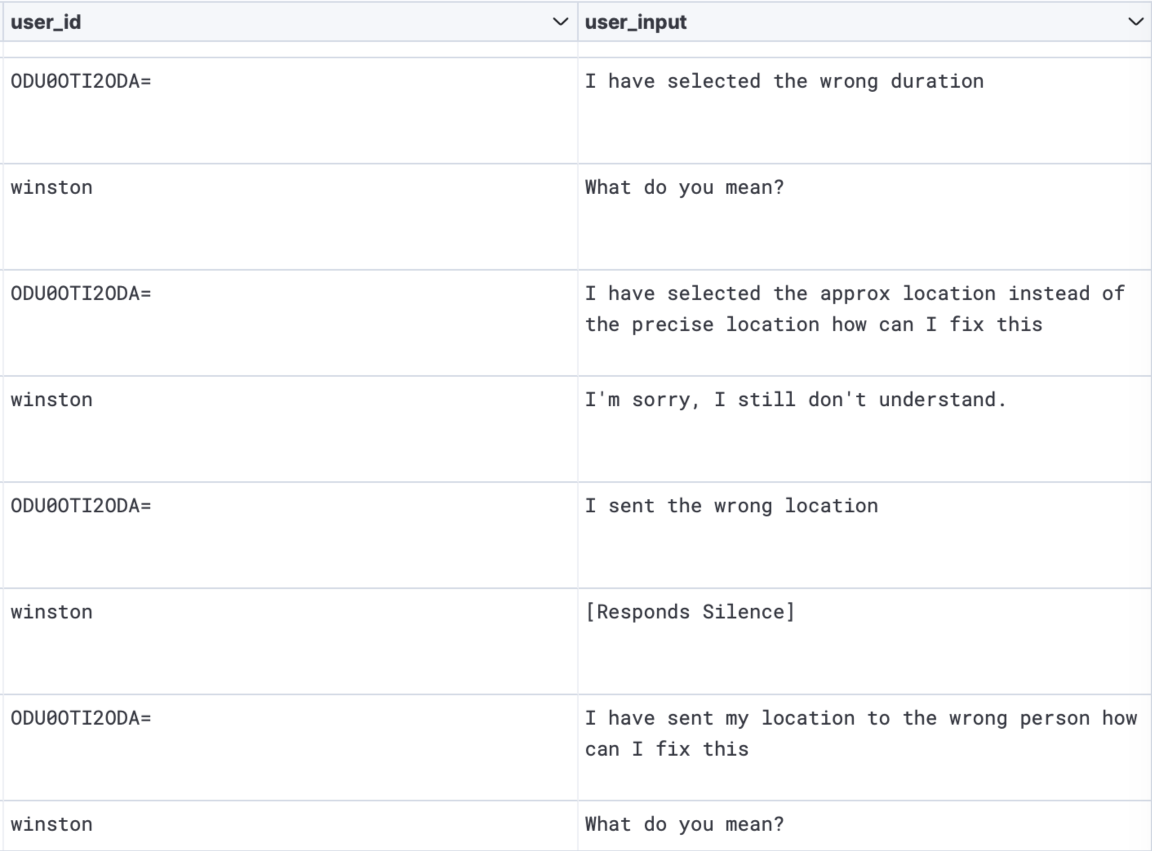


Figure 7-48: Distribution of whether a participant was able to resolve the error that they were given. More than half of the interactions were able to recover from errors.



user_id	user_input
ODU0OTI2ODA=	I have selected the wrong duration
winston	What do you mean?
ODU0OTI2ODA=	I have selected the approx location instead of the precise location how can I fix this
winston	I'm sorry, I still don't understand.
ODU0OTI2ODA=	I sent the wrong location
winston	[Responds Silence]
ODU0OTI2ODA=	I have sent my location to the wrong person how can I fix this
winston	What do you mean?

Figure 7-49: Interaction where exception handling intent fails. We can see that there is no clear indicator that an error has occurred. Later on, we see that the person does declare there was an error, but in the form of "[ERROR DESCRIPTION], how do I fix this"

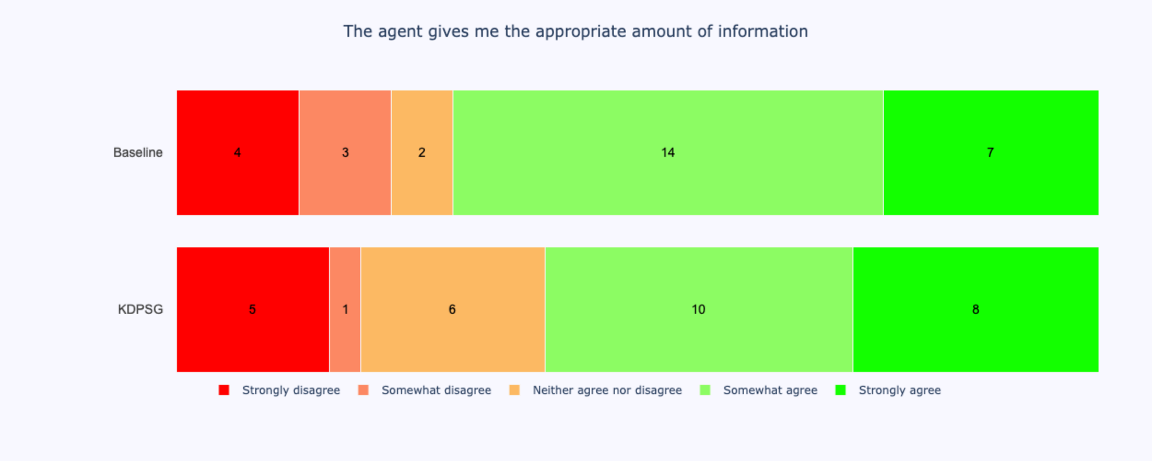


Figure 7-50: Distribution of whether an agent gave an appropriate amount of information or not. Participants leaned to agreeing that it was an appropriate amount of information.

are comparable. We tried to improve the intent detection for getting help for the original task by adding a few examples of failures that should have been handled, however both Study 1 and this one ran somewhat parallel, which limited the incorporation of the intent example improvements. We also see from the open response feedbacks that the intent detection for the error recovery needs to be improved. When we looked at the feedback of what went wrong in the error recovery, it is clear that we were once more heavily impacted by the intent detection and slot filling performance. The majority of the feedback on whether the agent misunderstood the user suggests that the agent struggled with understanding user input and frequently required users to rephrase or clarify their questions. Some users reported that the agent would forget previous conversation context, causing confusion. In the question of if participants were able to recover from the problem that was given, some interactions were able to recover from their mistakes relatively quickly and the agent gave them relevant information to fix the problem (33/60 interactions). However, other interactions struggled to communicate their mistakes to the agent and the agent did not understand them. In some cases, users had to start the conversation again, while in others, they gave up.

Interestingly, we examined the question of how would people ask a virtual agent

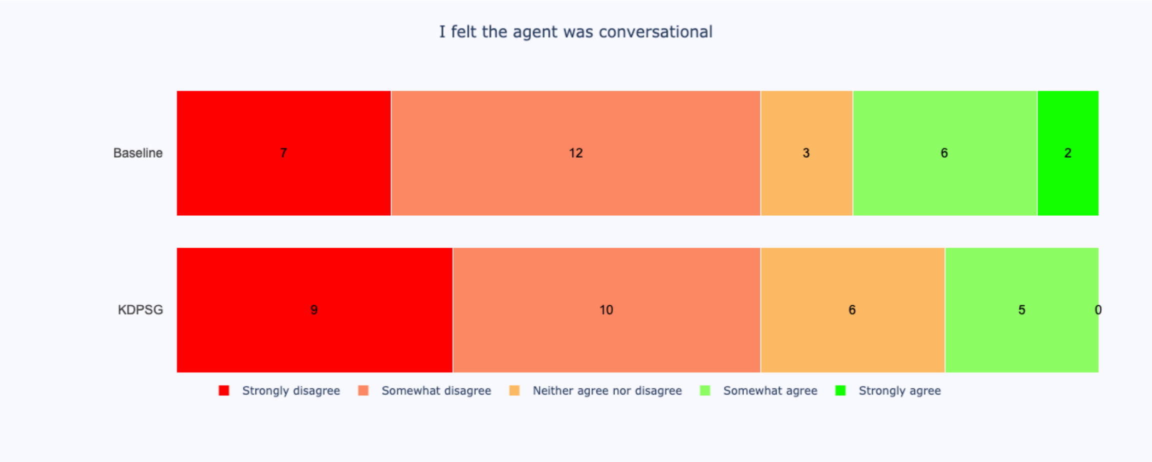


Figure 7-51: Distribution of whether a participant found the agent conversational or not. Most people thought that the agents were not conversational.

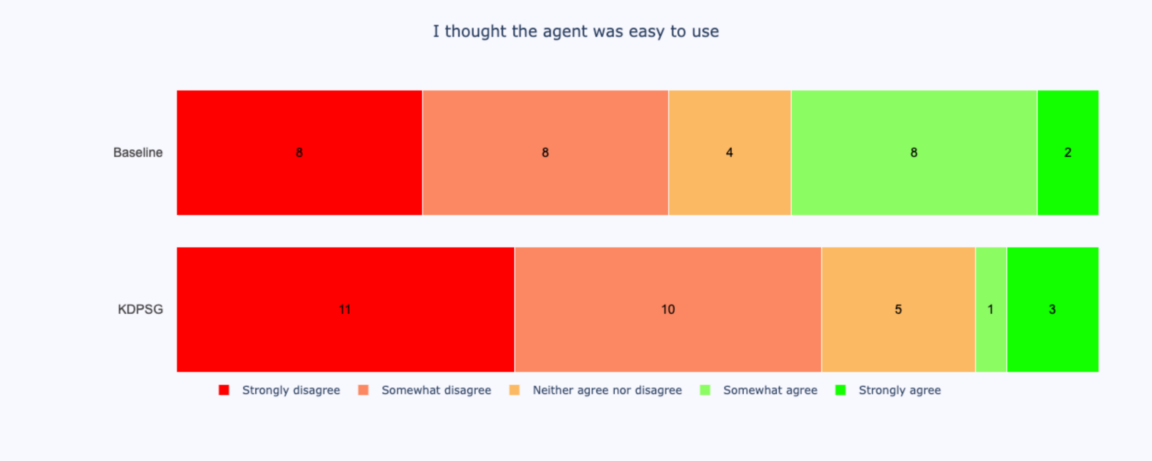


Figure 7-52: Distribution of whether an agent was easy to use or not. Most people did not find the agents easy to use.

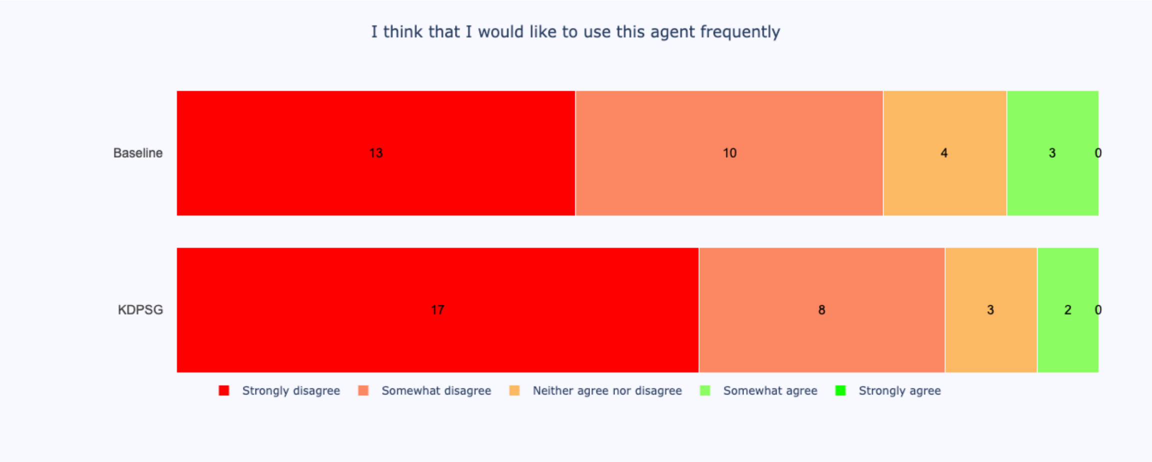


Figure 7-53: Distribution of whether a participant would use the agent frequently or not. Most people would not use the agents frequently

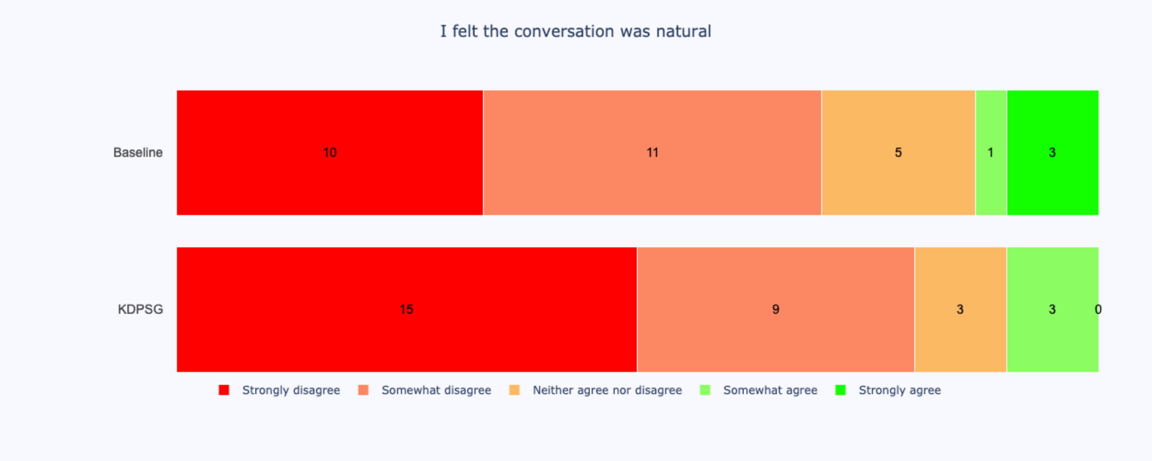


Figure 7-54: Distribution of whether the conversation felt natural or not. Most people did disagree that the conversation was natural.

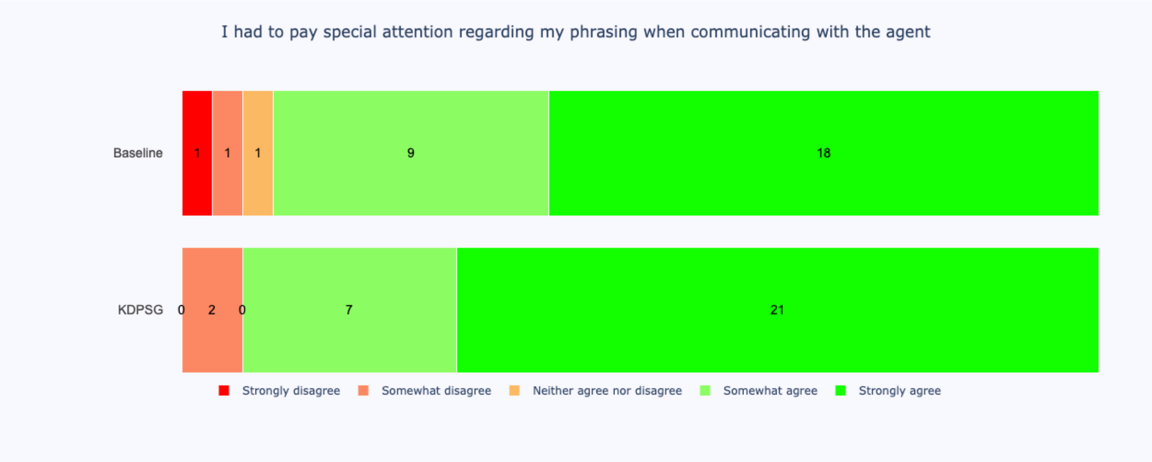


Figure 7-55: Distribution of whether a participant had to use special phrasing or not. Most people agreed that they had to use special phrasing with the agents.

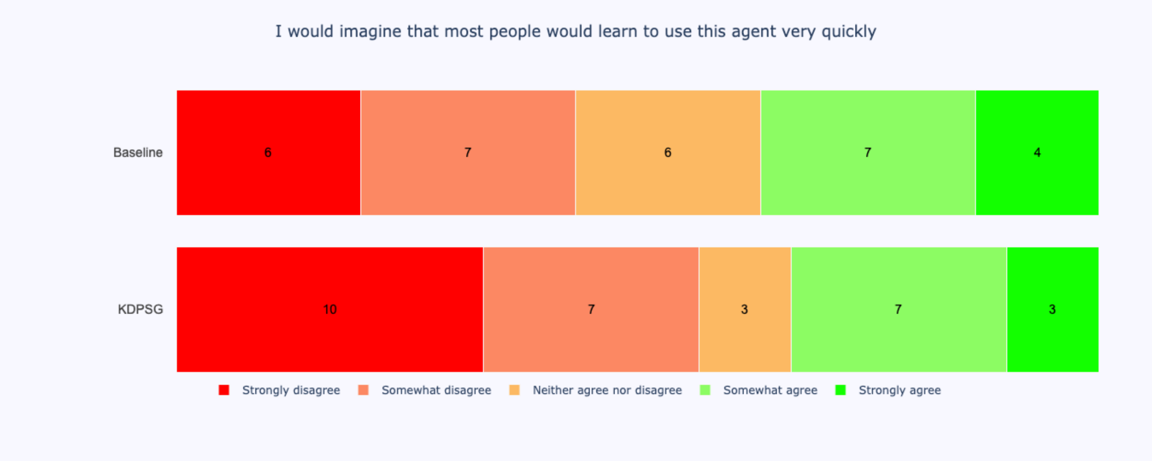


Figure 7-56: Distribution of whether participants would learn to use the agent quickly or not. Most people disagreed that others would learn to use the agents quickly.

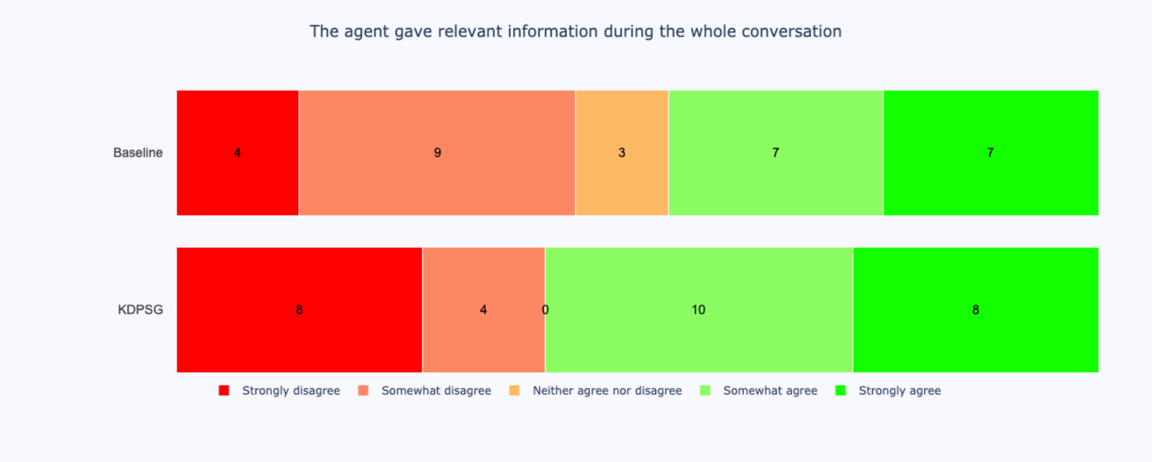


Figure 7-57: Distribution of whether an agent gave relevant information or not. Most people leaned towards that the agents gave relevant information.

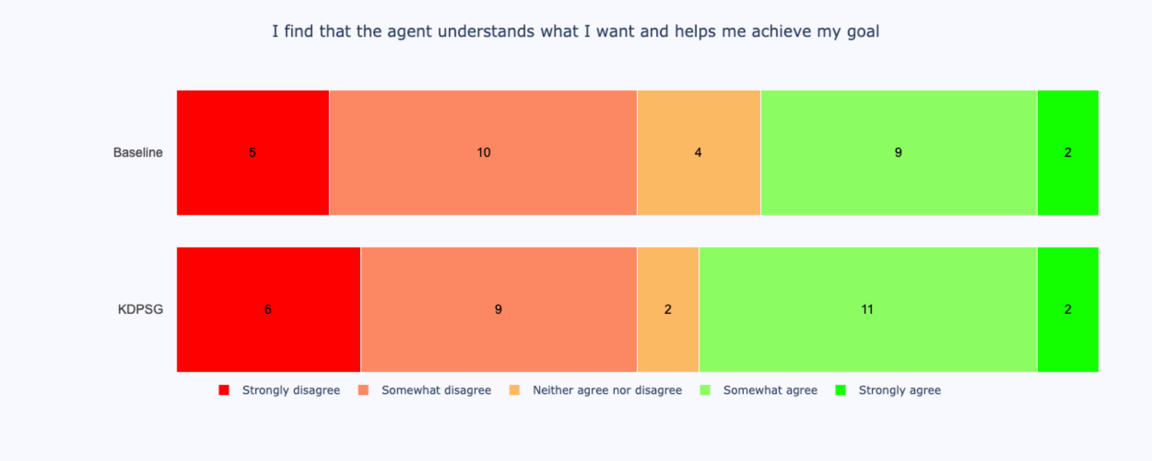


Figure 7-58: Distribution of whether participants thought that the agent understood them and helped them achieve their goals. Most people were neutral to negative that the agents understood them.

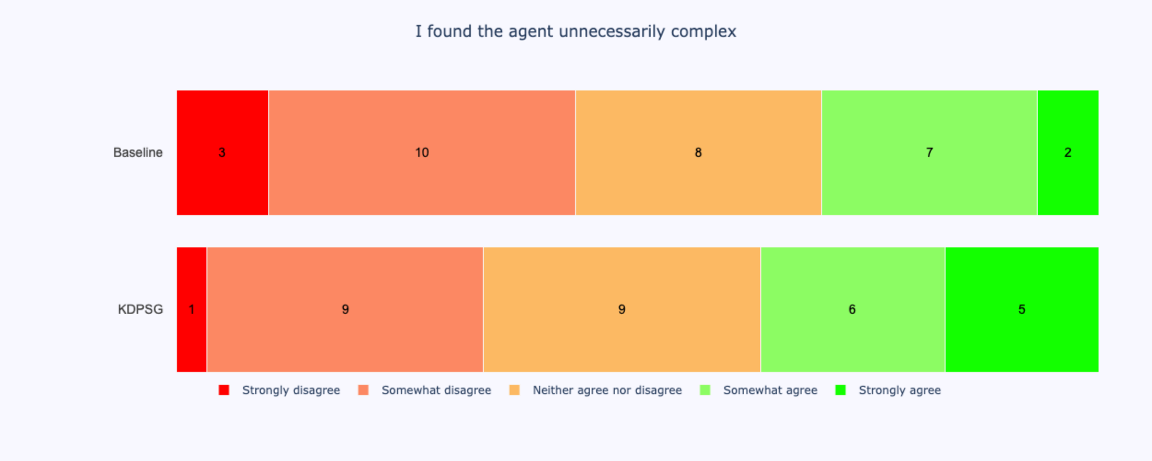


Figure 7-59: Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people were neutral on whether the agents were unnecessarily complex.

for help if they made a mistake and found the following. Some people would just ask for help directly, some would ask for instructions to be repeated, some would describe the mistake and expect the agent to understand this, and some would restart the process again. By looking at this feedback, we get some certainty that the way that we handled the exceptions in a procedure are justified, 19/60 feedbacks say they would declare that they made a mistake and describe it. However, a case that we did not anticipate is that some people would formulate the mistake as a question or declare what the mistake was (19/60 error recovery phrasing feedbacks). In the case that people describe what the mistake is, this is incredibly hard for an intent recognition system, and in the case that users framed it as a question, we tried to handle open-domain question answering as described in Chapter 6, however sometimes the intent recognition did not work and users would not have their question answered.

Finally, we looked at what participants felt worked very well when doing the procedure and when interacting with the agent. Overall, many users found the instructions to be clear, concise, helpful, and participants appreciated step-by-step guidance. Some users found that using specific language and keeping responses short and to the point helped the AI understand them better.

We note some quotes from what worked well during the interaction, and from people that were able to recover from the errors:

> “ I was able to recover from the problem relatively quickly and finished the original task. The agent gave me relevant information when it came to fixing the problem which allowed me to deal with the problem very quickly”
>
> “I feel that once the AI understood what was being asked of it, it explained how to perform the task well - instructions were incredibly clear and easy to follow.”
>
> “The agent provided simple and easy instructions”
>
> “The stepwise instructions were to the point and worked well at making the task easily comprehensible chunks”
>
> “The agent was very conversational and it felt like a real person.”
>
> “The instructions were very easy to follow and it felt like a real person was talking.”
>
> “Yes, I did recover but it took a while to communicate the mistake. ”

And some quotes of what did not work well in the interaction or from people that were unable to recover:

> “It was difficult to paraphrase/word sentences in such a way that would allow the AI to understand what I wanted it to do.”
>
> “Getting the agent to understand my mistake”
>
> “Understanding, lack of human abilities when it comes to memory”
>
> “I couldn’t get it to understand. It kept telling me how to change the timezone settings on my phone, which is not what I was asking. Eventually I gave up as it was just repeating the same information over and over again.”

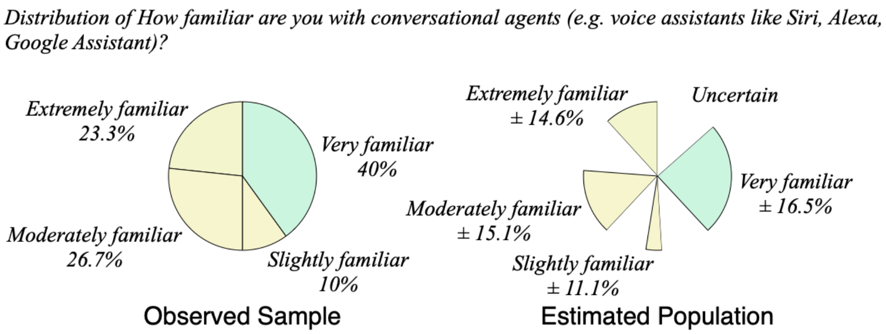


Figure 7-60: User distribution of familiarity with conversational agents. Most people were moderately familiar or more with agents.

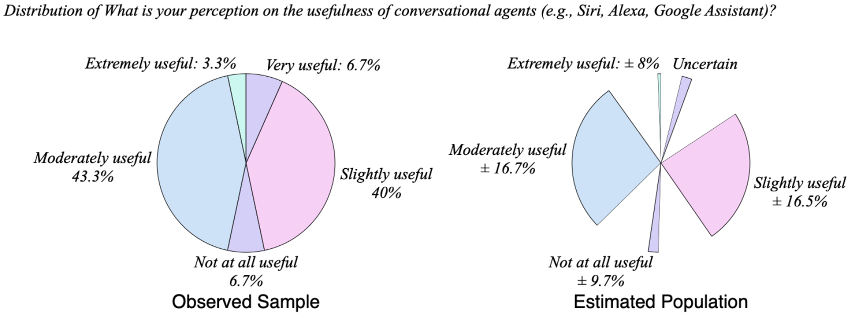


Figure 7-61: Distribution of participant's perception of general usefulness of conversational agents. Most people thought voice agents were moderately useful or more.

Golden Instruction Retrieval vs. GPT

We had 30 participants in our study (N=30). We now go through the results of our survey. We find that participants took on average 42.2 minutes to complete the survey. We find that most (N=27 or 90%) of the participants are moderately or more familiar with conversational agents, and that a minority of participants are slightly familiar with them. None of the users were unfamiliar of conversational agents. A view of this distribution can be seen in Figure 7-60. We also saw from the pre-study questions that most people (80%) find conversational agents to be moderately useful or more, as seen in Figure 7-61. Lastly, we see that most participants interact with a conversational agent at least once a day (70%) as seen in Figure 7-62. Now, in Figure 7-63 we report the reported performance of tasks.

We see, similar to the first study, that most tasks, save for composing an email

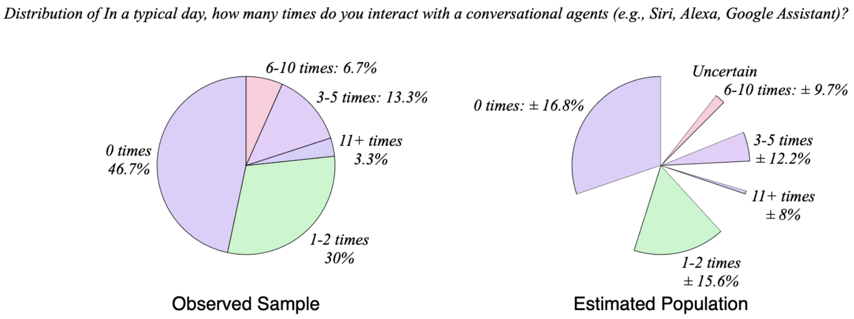


Figure 7-62: Distribution of participants reported interaction quantities with conversational agents. Most people interacted with agents 1 or more times, although almost half did not interact with them regularly.

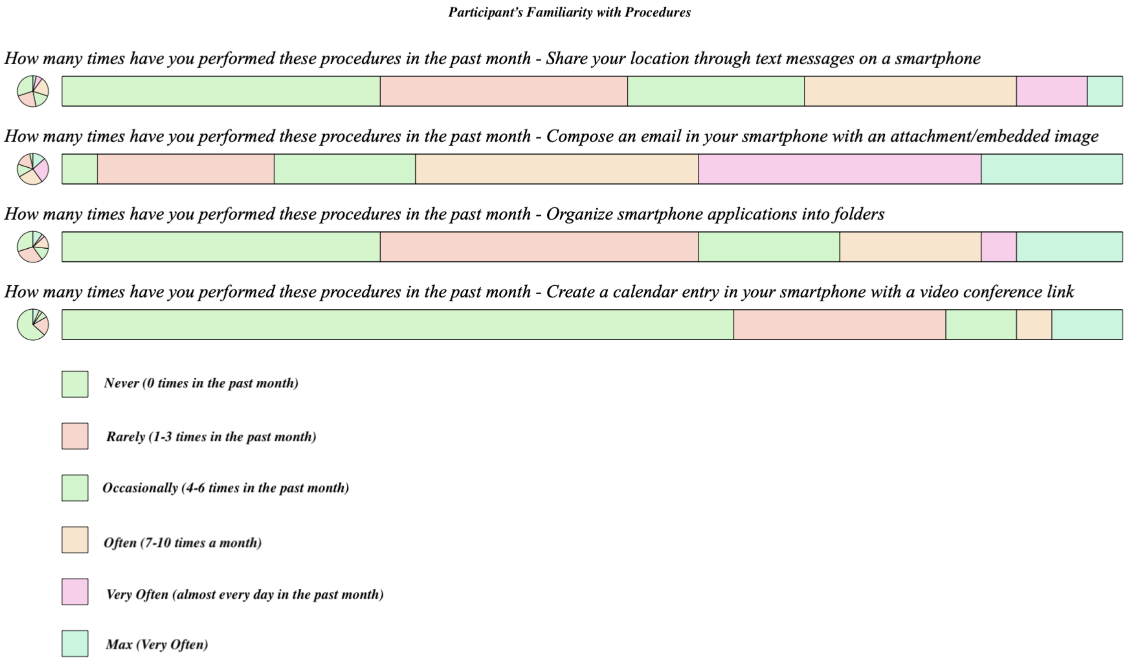


Figure 7-63: Distribution of participants reported monthly performance of tasks. We note that in this sample of participants, many had not made a calendar invite with video, and many (as in the other studies) had composed an email with an image attachment.

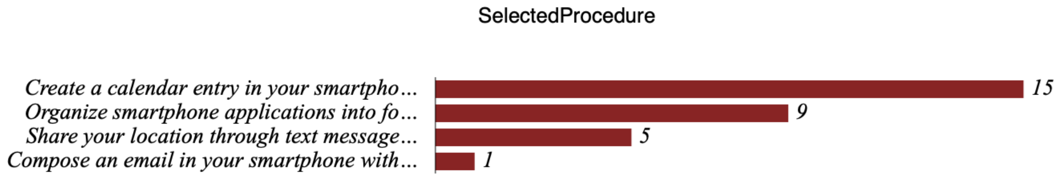


Figure 7-64: Distribution of the procedures that were assigned to participants. We note that from the reported procedures, the one that people had done the least is composed the calendar entry with a video link, so our selection strategy picked this one the most.

with an embedded image, have been performed "rarely (1-3 times)" or less in the past month. In this sample of participants, we also note that the amounts that the task were performed were higher than in the first study. Overall, we note that our selection of tasks was decent, with 3/4 tasks that people do not perform frequently. From this reported performance, we then sampled 1 task randomly from the set of least performed tasks. The distribution of tasks that a participant was given can be seen in Figure 7-64. We note that since users reported the email task as the most popular/familiar, this was the task that was sampled the least (1/30 participants were given this task).

When then moved on to review the feedback of whether the tasks that were given were appropriate or not for a conversational agent. Most participants (26/30) agreed that the tasks were simple and straightforward, making them suitable for a virtual assistant to provide step-by-step instructions. Some participants also mentioned that the tasks were tech-related, making them ideal for an AI assistant to handle. However, some participants were skeptical, stating that they would prefer to do the task themselves or that the AI assistant might not be able to handle more complex tasks.

Now we move on to the more important results from the study. These can be seen in Figures 7-65, 7-66, 7-67. From here, we can see that a significant portion of participants (29/30 for GPT and 27/30 for the gold instruction baseline) were able to follow the instructions that an agent gave them and that the majority (29/30 for

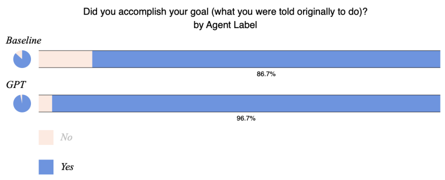


Figure 7-65: Distribution of whether a participant was able to accomplish the goal they were given or not. Most interactions were able to complete their goals.

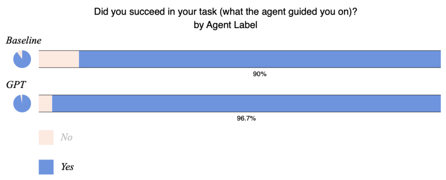


Figure 7-66: Distribution of whether a participant was able to accomplish the instructions that the agent gives. Most interactions were able to follow the agents' instructions.

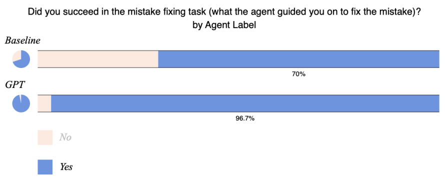


Figure 7-67: Distribution of whether a participant was able to resolve the error that they were given. We note that almost everyone was able to recover from errors (70%+) for both agents, and all but one recovered with the GPT based agent.

the LLM (ChatGPT) agent and 26/30 for the gold instruction baseline agent, both of which are significantly higher than a 50% success rate with a binomial test ($p<0.05$)) were able to complete the original that they were assigned. Now, the most important of results of this study can be seen in Figure 7-67, which is that for both agents, more than 70% of the participants were able to recover from the error that they were given, and in the case of the ChatGPT-based agent all but one were able to recover from the error.

Now, in the case of the baseline agent, there is a difference between the amount of people that were able to complete the original process (26/30) and the amount of people that were able to recover from the error (21/30). We then looked into the cases where these agents failed (1/30 for the GPT agent, and 9/30 for the Baseline) to examine what happened. In the case of the interaction that failed from GPT, the participant did not perform the error handling, because there was no option to change the duration on their phone. We then moved on to examine the 9 interactions that failed in the Baseline. We found that, similar to all other interactions that involved

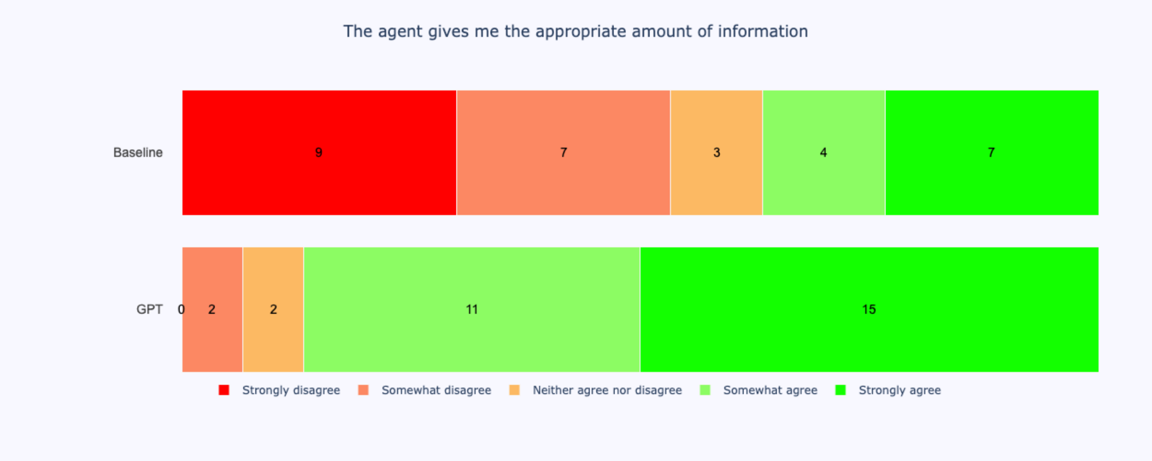


Figure 7-68: Distribution of whether an agent gave an appropriate amount of information or not. We see that participants found that the GPT agent gave more appropriate amounts than the baseline.

the intent based agent, most failed due to some intent matching/slot-filling issue (5/9 or 56% of the interactions that failed). The ones in which the intent matching did not fail, the users were unable to understand the instructions given, or had some other difficulty completing the steps. Once more, as in the other block in this study, in the open text question of what they would do if they made a mistake that many people would explain the mistake that was made and ask for specific and clear instructions on how to fix it or move forward. Once more, we see that our approach of trying to catch the intent by a declaration of a mistake is acceptable, but it does not cover the case in which the person only describes what went wrong. Next, we looked into the open text feedback given from the survey question: "Were you able to recover from the problem that was given ([ERROR]) and finish the original task? Describe any issues or things that went well". Here most people mentioned that they were able to fix the error, and that in the case of the ChatGPT-based agent, people praised it answered in simple, concise responses. In the case of the baseline agent, some people had to rephrase to get the system to understand, but were eventually able to fix the issue. Now we move on to the usability questions.

We once see that there were significant usability issues with the baseline agent. In all usability dimensions that we evaluated, the GPT agent had significantly different

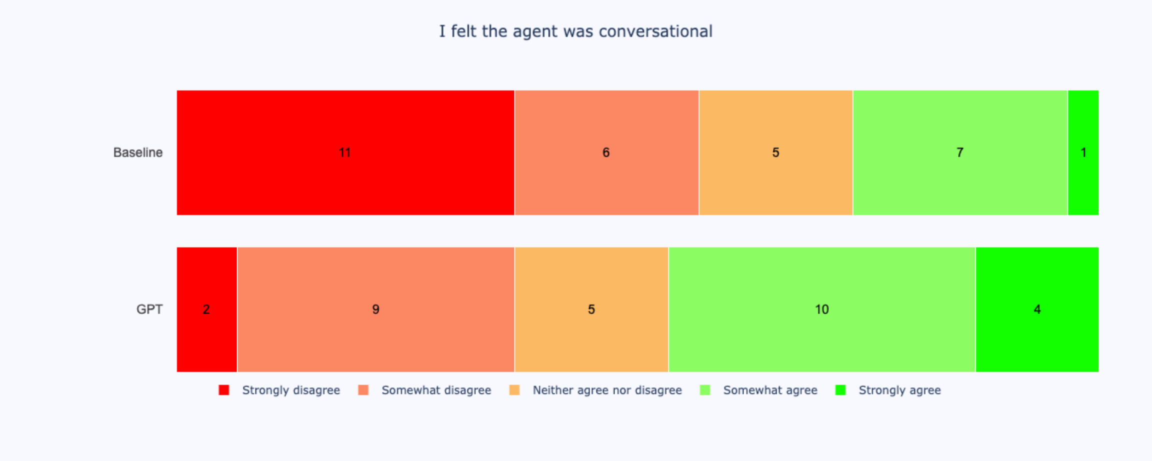


Figure 7-69: Distribution of whether a participant found the agent conversational or not. Most participants were neutral on whether the GPT agent was conversational, and people disagreed that the baseline was conversational.

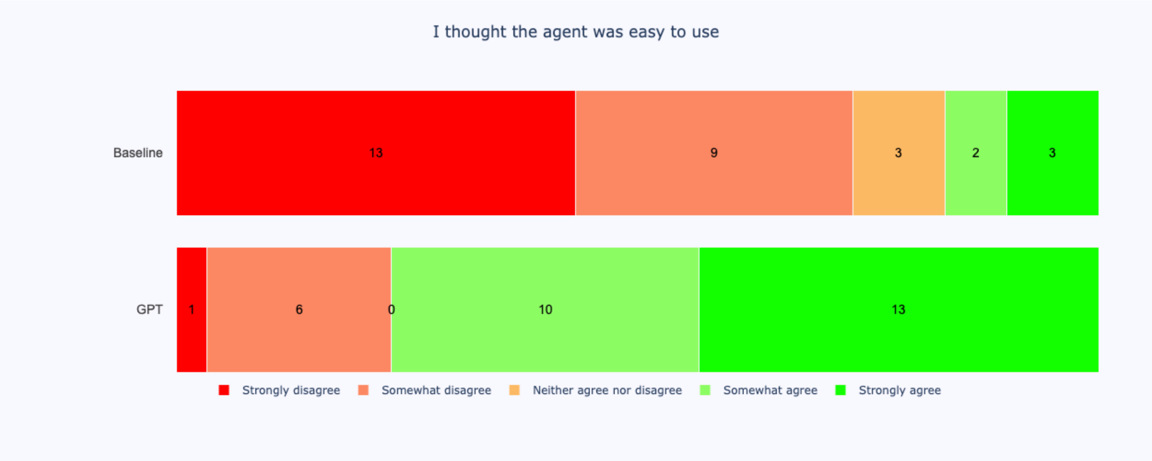


Figure 7-70: Distribution of whether an agent was easy to use or not. Most people leaned towards agreeing that the GPT based agent was easy to use in contrast to the baseline which was seen as not easy to use.

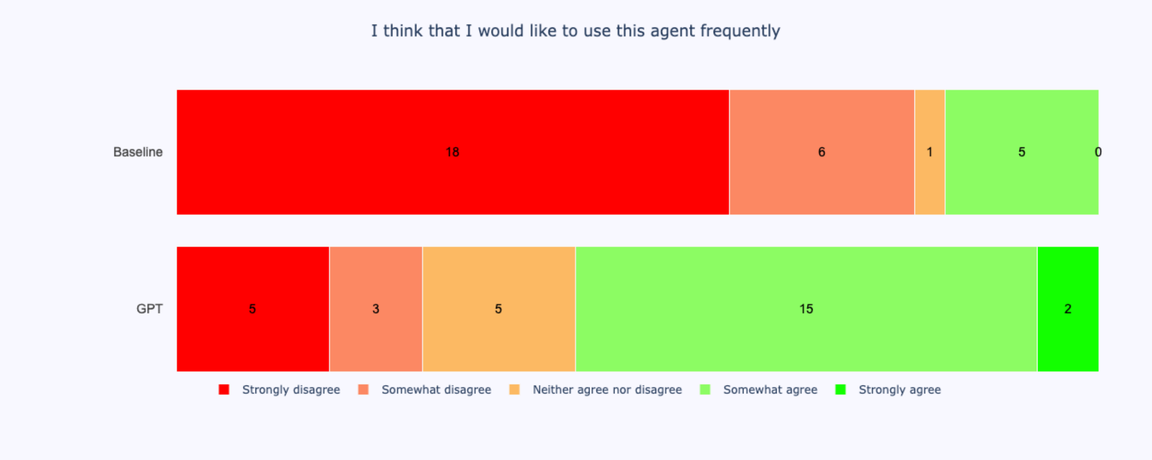


Figure 7-71: Distribution of whether a participant would use the agent frequently or not. Most people were neutral on whether they would use the GPT-based agent in contrast to the baseline which people would not use frequently

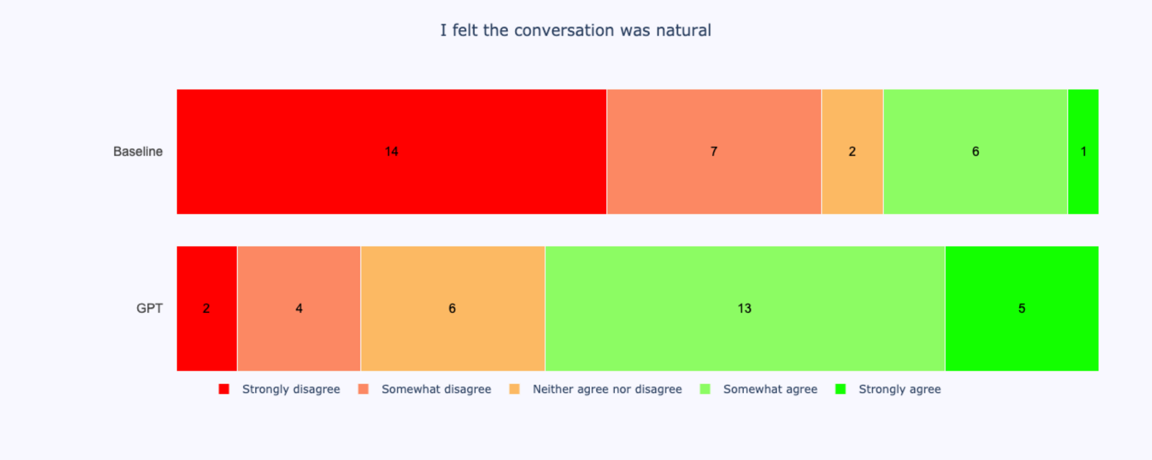


Figure 7-72: Distribution of whether the conversation felt natural or not. Most people felt that the GPT-based conversation was more natural than the baseline conversation.

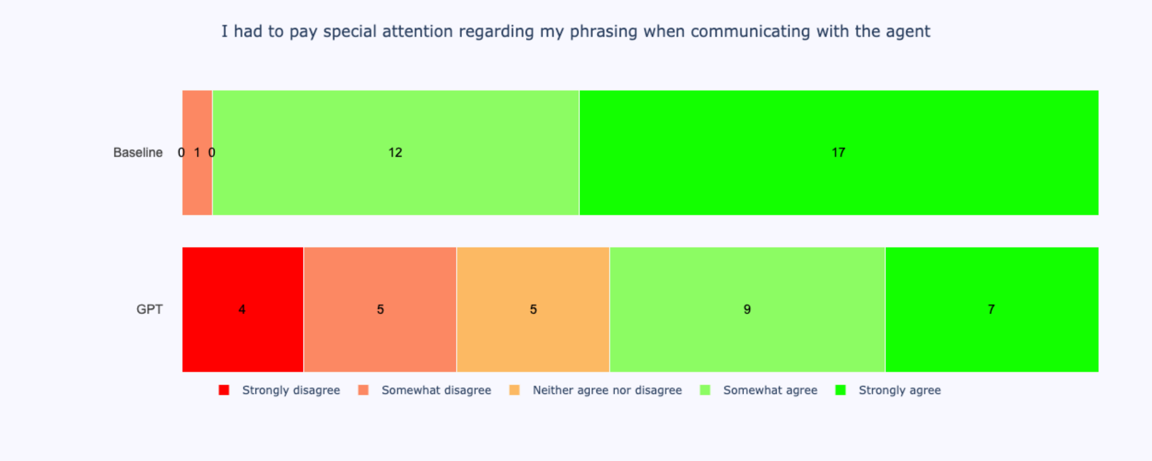


Figure 7-73: Distribution of whether a participant had to use special phrasing or not. Most people found that they needed to use special phrasing with the baseline and were neutral with respect to the GPT agent.

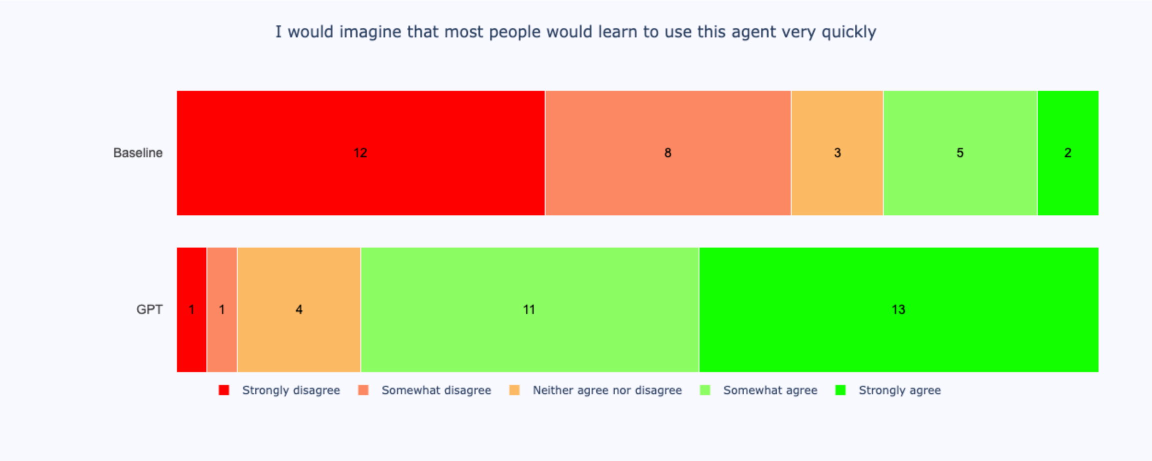


Figure 7-74: Distribution of whether participants would learn to use the agent quickly or not. Most people thought the GPT agent would be quicker to learn to use than the baseline agent.

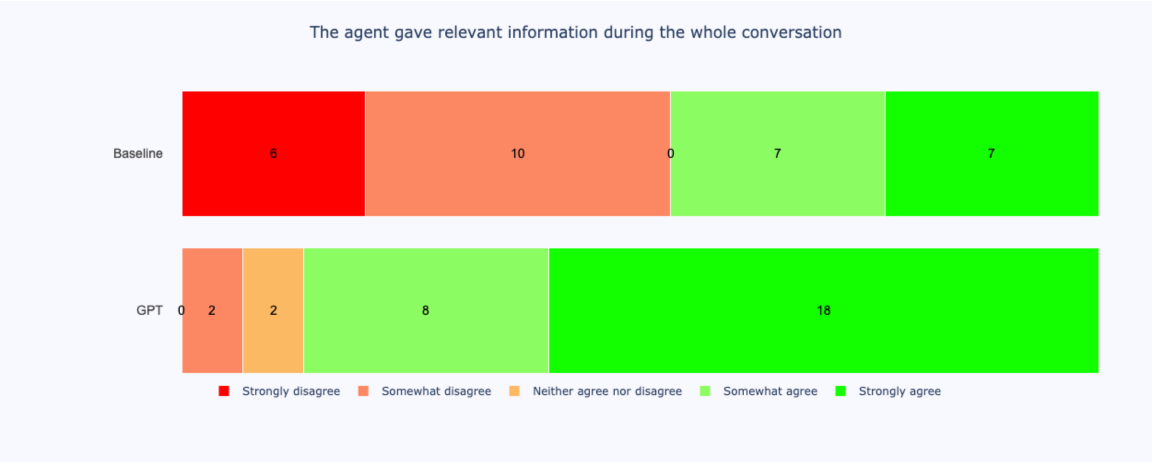


Figure 7-75: Distribution of whether an agent gave relevant information or not. More people felt that the GPT agent gave relevant information and were neutral on the baseline agent.

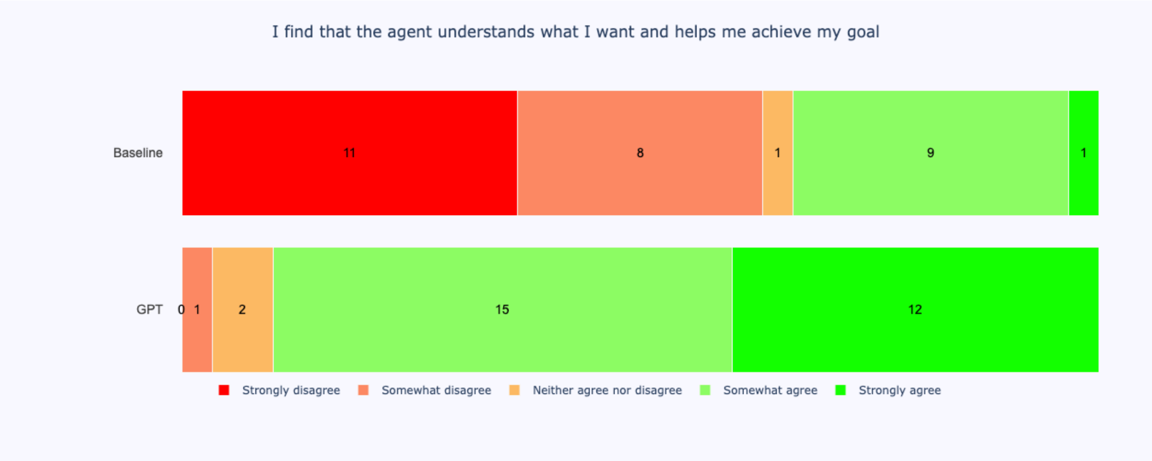


Figure 7-76: Distribution of whether participants thought that the agent understood them and helped them achieve their goals. Most participants felt that the GPT agent understood them better than the baseline agent.

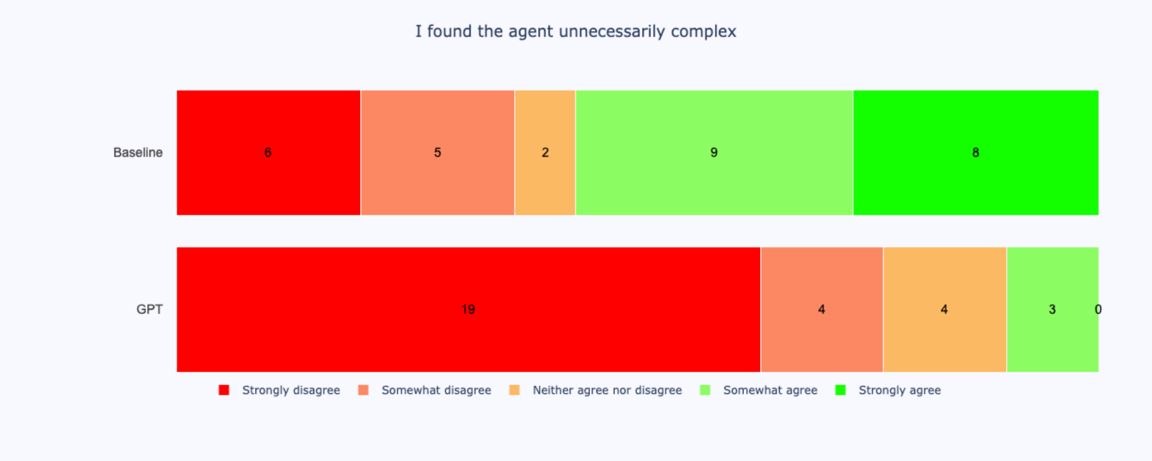


Figure 7-77: Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people disagreed that the GPT-based agent was complex, and were neutral on the baseline.

results ($p<0.05$ with a Wilcoxon signed rank test). When we looked into the open text responses for things that did not go well in the interaction or with the task, we see that participants were frustrated with the baseline agent's ability to understand instructions and questions, as well as some comments about poor voice recognition and transcription, and the need for very specific questions to get to the end goal. We saw that in the feedback for the ChatGPT based agent, people praised being able to communicate with the agent more naturally, but sometimes wanted to go step-by-step rather than the long responses that the system gave.

Some quotes on what people found that worked well in the interaction and about people that were able to recover successfully are the following:

> "The ease and clearness of the instruction"
>
> "instructions were very accurate and easy to follow, did not feel lost for a single second"
>
> "Everything. In both the original task and the recovery from the mistake the agent answered in one efficient message."
>
> "Yes, it was very easy, only a small change that the agent told me to do immediately."

Finally, some quotes about what did not work well in the interaction and from people that were unable to recover:

> “The agent struggling to understand, and the voice dictation”
>
> “The agent was slow to explain each step, I would appreciate more haste.”
>
> “The instructions were too basic and slow, especially the zoom link one. Also the agent got confused easily”
>
> “No. He couldn’t understand I what I said no matter how many times I tried to paraphrase.”

We note here that in most of these studies, at least 1 person complained of the use of continuers in the process. Especially for the intent-based agents. One possibility for this is that the instructions may have been too simple for the user (as was noted in the quote), or that different users prefer different amounts of information at once (as was seen in a quote in a prior study). Nevertheless, for future work it should be taken into consideration the ability to either give all the steps at once or to not require continuers (set a timer to move the conversation forward and give another step after a certain time).

In a conversation-first system, producing a continuer can be faster than waiting for the next instruction. However, in a voice+visual first system, continuers may be bothersome as the person can read the screen to get the step. Future work should look into evaluating a voice-only interface to better demonstrate the value of the extended telling pattern, where users cannot just look back at the list of instructions in the chat history and would need to let the system know when to continue or go back/forth.

7.3.5 Overall Findings

We now list the findings from our study.

- Our evidence shows that technical intent-based / slot filling issues aside the systems we evaluated (Intent based Gold-instruction retrieval KDPSG and

ChatGPT-based system) *can* provide guidance in error recovery situations [H5].

- Our evidence also shows that there is no significant difference in error recovery guidance between the baseline gold instruction retrieval system and our knowledge driven instructions [H6].

- Our evidence suggests that guidance in error situations has to be flexible enough to understand first that there was an issue in the step, which may be possible with an LLM-based model, but may be hard with an intent-based system [H8]

- Our evidence suggests, that maintaining context in these error interactions may be essential, as sometimes people referred to things in the prior steps and asked to change parts of it or framed their mistake in the context of prior steps [H8]

- Our evidence shows once more, that our baseline and knowledge driven agent were impacted negatively in usability to the point that they are not natural nor conversational because of the intent matching and slot filling capabilities and further improvement is needed [H7]

- Our evidence shows that the LLM based agent was much more flexible in error handling, and could easily provide guidance in an exception process, leading to a more natural and conversational system [H7]

- Our evidence suggests that sometimes people preferred the step-by-step instructions rather than a single long answer [H8]

- Our evidence shows that step-by-step instructions, especially if they are clear and concise, can help guide people in procedures and out of errors [H8]

- Our evidence suggests that some users may want to get all instructions at once or not have to give continuers to get the next step. This behavior should be supported, whether through inquiries or timers, to satisfy users. [H8]

7.4 Evaluation 3: Leveraging both flexibility of LLMs and knowledge driven steps

7.4.1 Overview

So far, we have seen that our intent-based knowledge driven procedural step generation system, is lacking severely in the flexibility that the intent recognition and slot filling allow. We also saw that the ChatGPT-based agent did not suffer from these flaws, but that it was possible for the system to hallucinate if the information it has to give is outdated or if it is a niche/expert domain. In this final evaluation, we test whether supplying the ChatGPT powered agent with the up-to-date, knowledge driven steps and some facts, can leverage both the flexibility of the LLM models and the ability to from knowledge (possibly updated) to generate procedure steps. Our results show that indeed this hybrid, retrieval-like approach can inform the LLM model in such a way that people find it to provide more relevant, accurate, and reliable information, than without it.

7.4.2 Research Questions

The explicit questions that we set out to solve with this experiment are the following:

1. H9: Can a LLM conversational agent combined with our knowledge driven system and description of conversational patterns be more accurate and reliable than a plain LLM system?
2. H10: Is this agent natural and conversational?
3. H11: Can we get some design feedback on our procedural guidance agent?

7.4.3 Study Design

In our third study, we utilize a similar format as the first evaluation in Section 7.2.3. Since we wanted to evaluate the performance of the agents in possibly outdated or unknown procedures, we had to change the procedures that we gave to users. We came up with the following set of tasks:

1. How to unsend a text message on iPhone
2. How to edit a sent message on iPhone
3. How to change the clock style on the Lock Screen on iPhone
4. How to share location on messages on iPhone

We note that we selected the procedures for iPhones, and controlled for this in the Prolific settings, because it was simple to trace the dates for operating system releases, most users consistently update to the latest version of iOS, and the experience is almost uniform throughout a wide range of devices. We note that as of the time of this writing, the training data with which ChatGPT (our LLM agent) was trained on (ca. 2021), would show that the first three of those procedures were impossible to do on iPhone. The fourth procedure, changed (there used to be an "i" icon in message threads, which is now a video icon).

Again, we opted to perform this study with the Prolific[4] platform and use URL query parameters to generate links that participants can access different agents with. We also use the same within-subjects design [13] to maximize the population of the study.

In the process, we first asked users to rate their frequency of performing the tasks that were listed above on a Likert scale of 1-5, with 1 indicating they had not performed the task in the past month and 5 indicating they performed the task almost daily. From the tasks that users had performed the least, we randomly selected one and had the user ask for guidance on how to perform the task using a voice agent. We had the user perform the task twice and provide feedback each time, with the guidance of a different agent. We conducted the study in blocks of two, with the baseline agent (plain ChatGPT) compared against the other agent, the hybrid (ChatGPT+KDPSG prompt). We note that we indicated that users should try to utilize the microphone as much as possible, but that they were free to utilize the keyboard and write to the

---

[4] `https://www.prolific.co/`

agent if there were issues. A difference from the first procedure is that we include the following factuality/reliability questions:

- I think the guidance I was given is up-to-date (Yes/No)
- I feel like the agent's responses were accurate. (5 point Likert)
- I believe that the agent only states reliable information. (5 point Likert)
- It appeared that the agent provided accurate and reliable information. (5 point Likert)

We also asked the following questions about procedural interactions:

- What kinds of procedures do you think these conversational agents would be helpful for? (Open response)
- What kinds of procedures do you think these conversational agents would NOT be helpful for? (Open response)

7.4.4 Study Results

We now discuss the primary results of the study. In Figures 7-79 and 7-78, we see that people with the Hybrid Knowledge Driven + GPT (KGPT) had a significantly higher ($p<0.05$ for both χ^2 and Fisher's exact test) success rate than with the plain GPT. Similarly, there is a significant difference ($p<0.05$ for χ^2 test) for being able to complete their goals when using the GPT vs. the KGPT agent. We see then that on the questions relating to the accuracy and reliability of the information given by an agent, that there is a significant difference ($p<0.05$ with a Wilcoxon signed-rank test) in Figures 7-80,7-81,7-82. Lastly, we see that in Figure 7-90, that there is a significant increase ($p<0.05$ with a Wilcoxon signed rank test and t-test) in people's agreement of the agent giving relevant information for the KGPT agent.

Altogether, these results show that the inclusion of the outputs of the KDPSG system (both facts and steps) helps the agent provide more accurate and reliable information than without these. Although we do not explore other methods of retrieval, this test is a good indicator that retrieval, whether based on the KDPSG, or

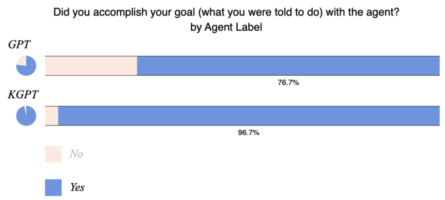


Figure 7-78: Distribution of whether participant were able to succeed in their goals or not. Most interactions were able to succeed in their goals, but the ones with the knowledge driven hybrid had higher success rates

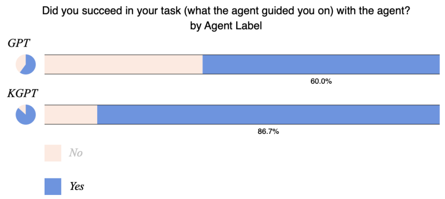


Figure 7-79: Distribution of whether participant were able to succeed in what the agent told them to do. We see once more that most interactions were able to succeed in what the agent guided them on, but the knowledge driven hybrid had a higher success rate of participants being able to follow its instructions.

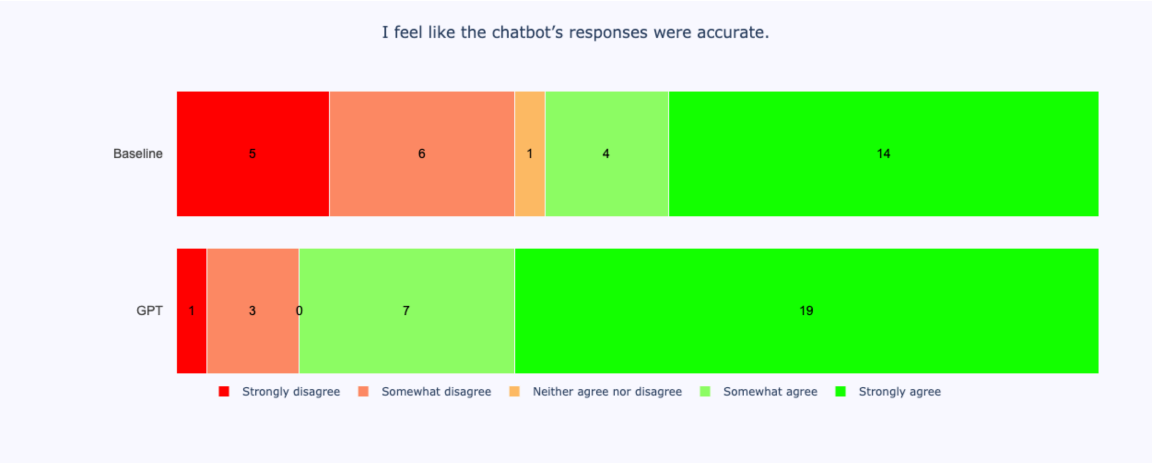


Figure 7-80: Distribution of whether participant thought the agent's content was accurate. On the baseline ChatGPT agent, people were neutral leaning to agree, but there was a significantly higher agreement that the hybrid approach was more accurate

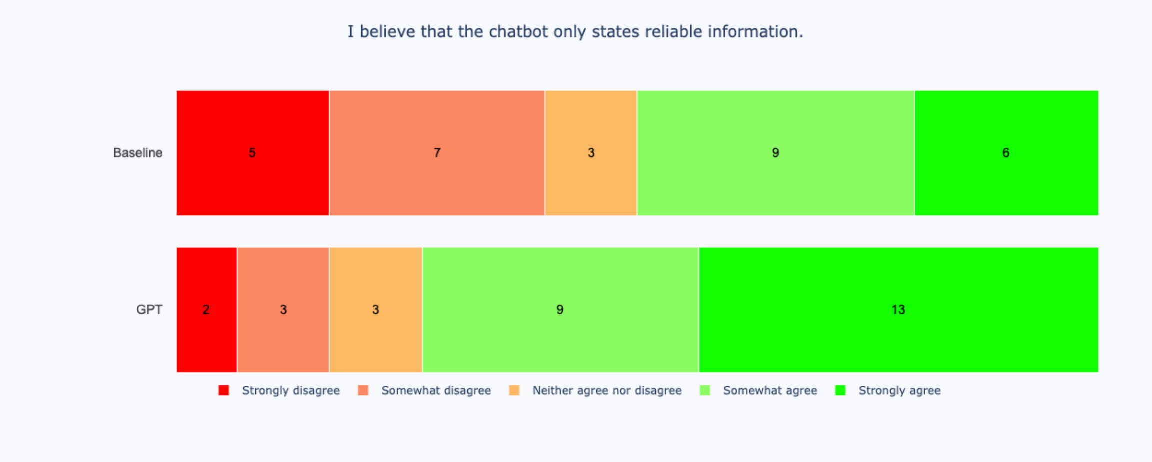


Figure 7-81: Distribution of whether participant thought the agent's content was reliable. On the baseline ChatGPT agent, people were neutral leaning to agree, but there was a significantly higher agreement that the hybrid approach was more reliable

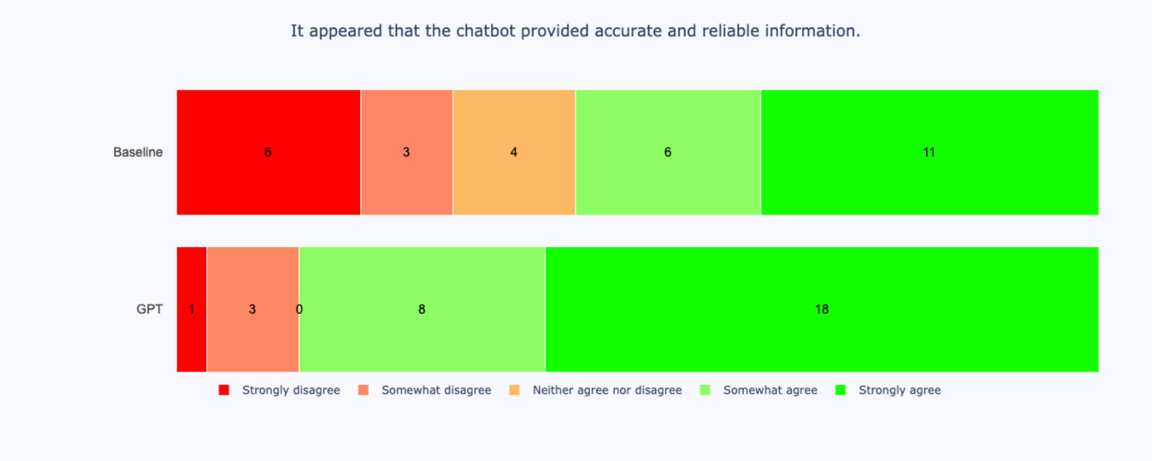


Figure 7-82: Distribution of whether participant thought the agent's content was accurate and reliable. On the baseline ChatGPT agent, people were neutral leaning to agree, but there was a significantly higher agreement that the hybrid approach was both more accurate and reliable

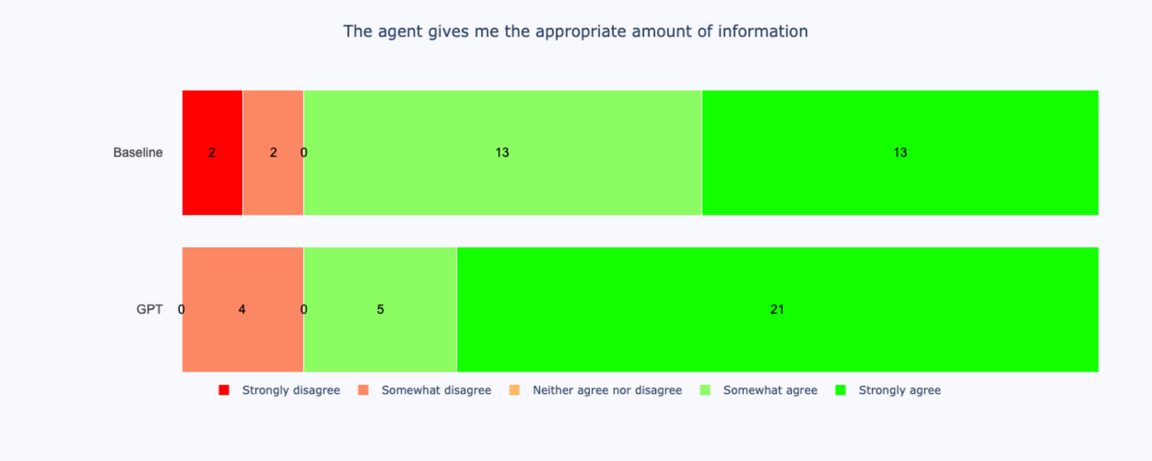


Figure 7-83: Distribution of whether an agent gave an appropriate amount of information or not. We see that participants found that the GPT agent gave more appropriate amounts than the baseline.

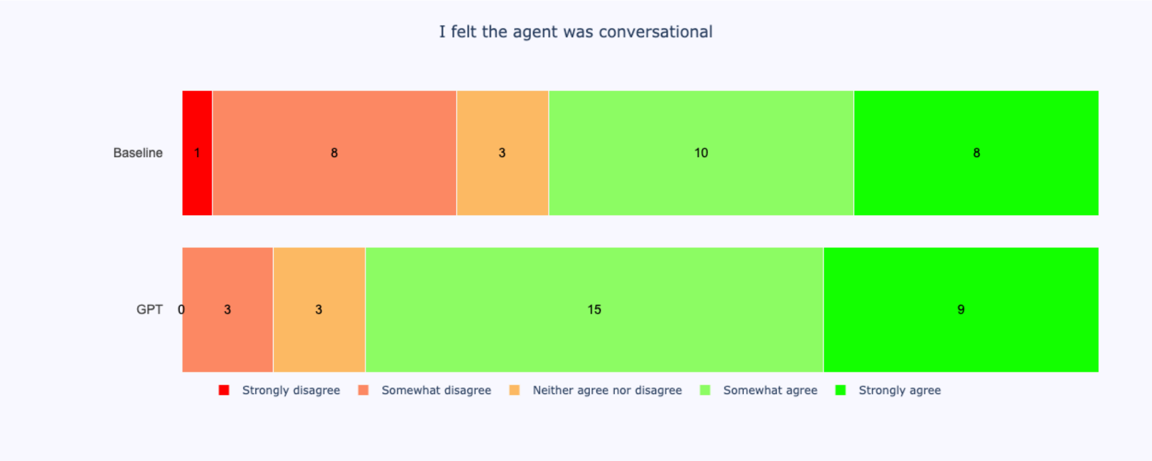


Figure 7-84: Distribution of whether a participant found the agent conversational or not. Most participants were neutral on whether the GPT agent was conversational, and people disagreed that the baseline was conversational.

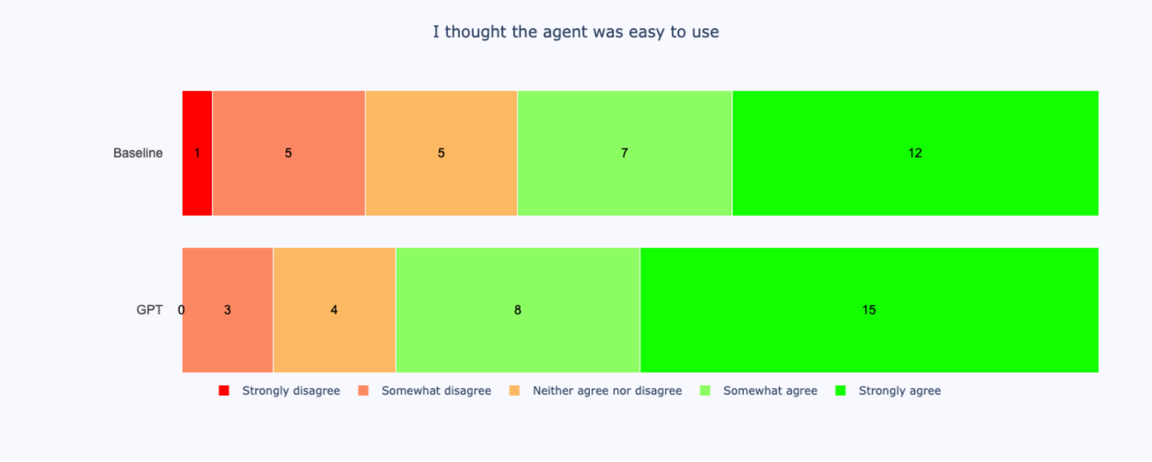


Figure 7-85: Distribution of whether an agent was easy to use or not. Most people agreed that both agents were easy to use.

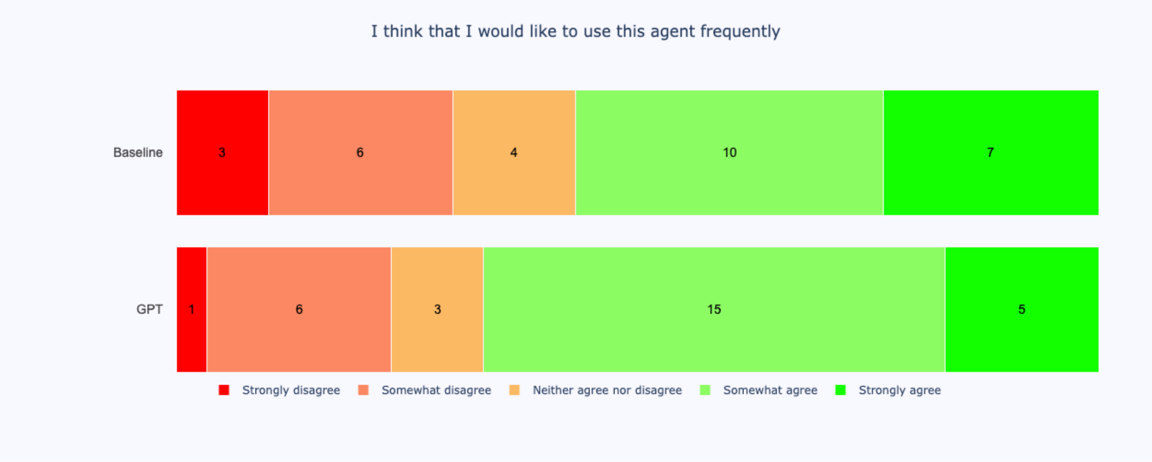


Figure 7-86: Distribution of whether a participant would use the agent frequently or not. Most people thought that they would use both agents frequently

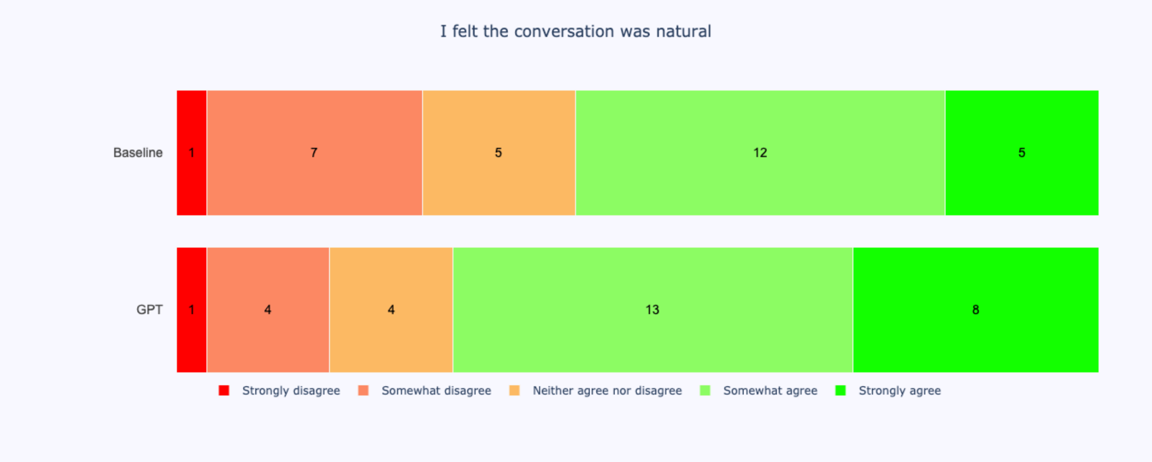


Figure 7-87: Distribution of whether the conversation felt natural or not. Most people felt that both agents were somewhat conversational

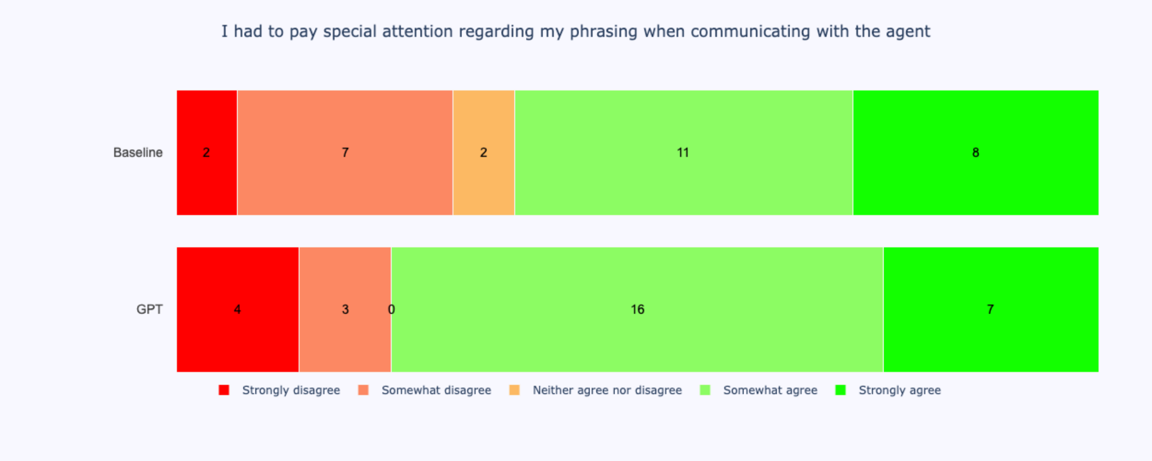


Figure 7-88: Distribution of whether a participant had to pay special attention to phrasing or not. People were neutral on whether special attention phrasing needed to be used, and slightly more agreed that it was needed on the Hybrid system.

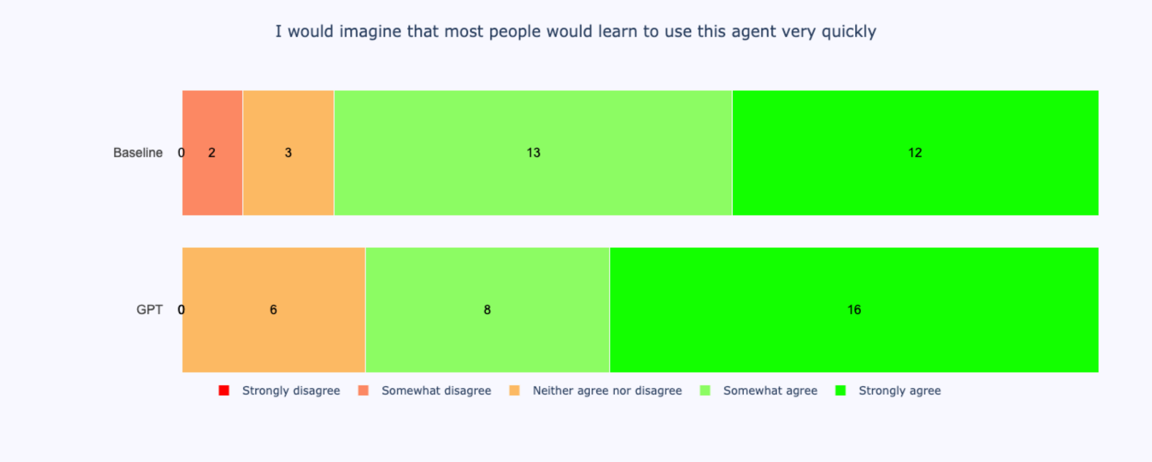


Figure 7-89: Distribution of whether participants would learn to use the agent quickly or not. Most people thought that both agents could be learned to use quickly.

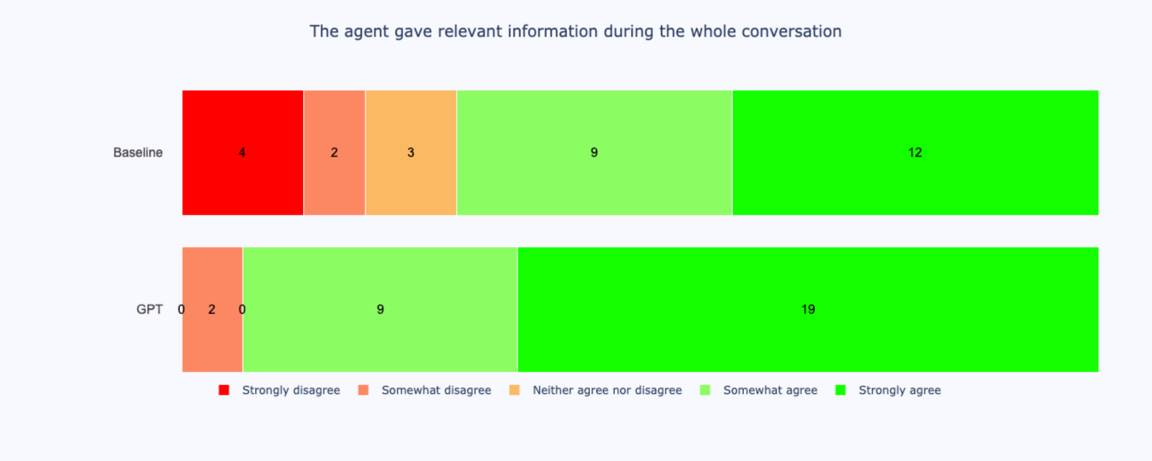


Figure 7-90: Distribution of whether an agent gave relevant information or not. More people felt that the Hybrid agent gave relevant information than the plain ChatGPT based agent.

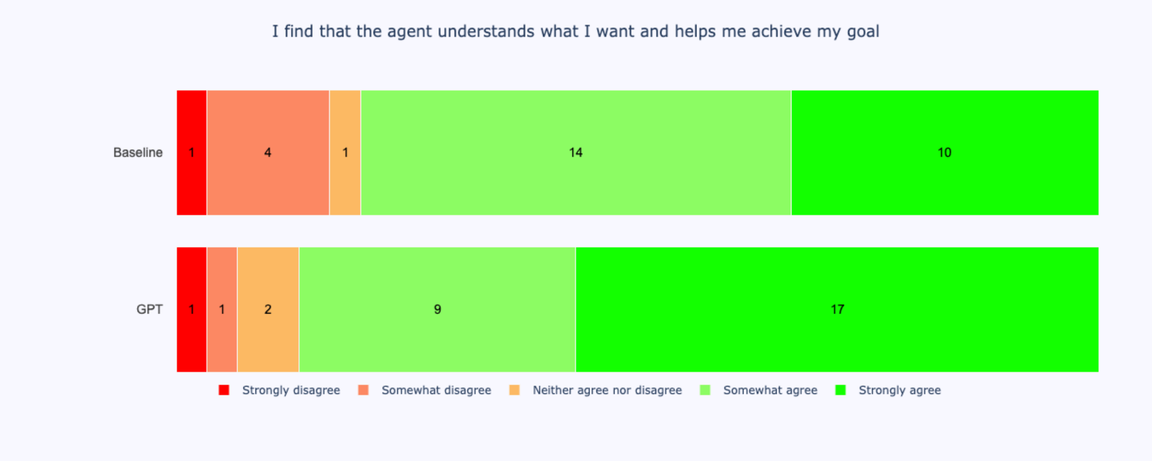


Figure 7-91: Distribution of whether participants thought that the agent understood them and helped them achieve their goals. Most participants felt that both agents understood and helped them achieve their goals.

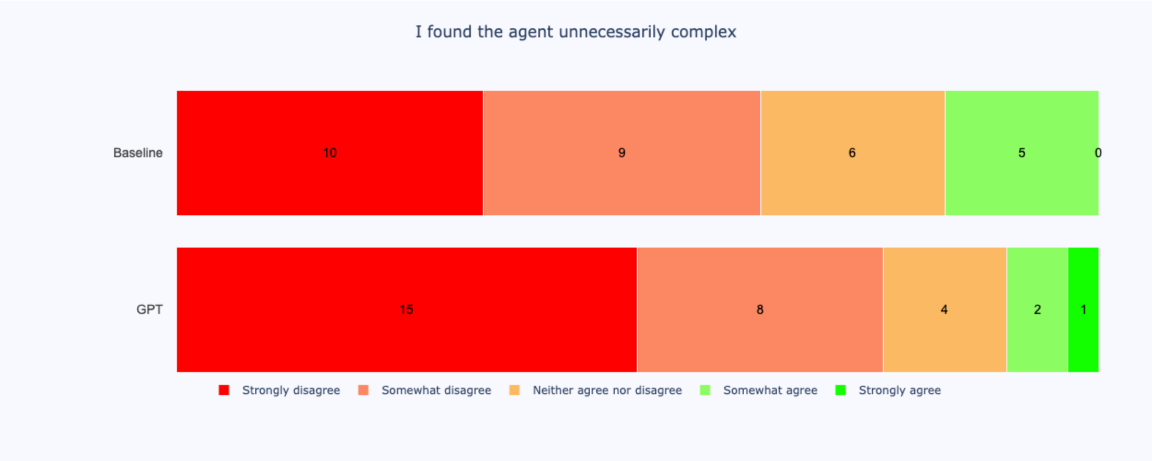


Figure 7-92: Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people agreed that both systems were not complex, with slightly more agreeing on the Hybrid system.

other methods, is an interesting route for future systems in the area. We also note that this may be applicable in domains where the procedures may often change (e.g., software-related procedures), although through user feedback, we can see that it may be applicable to expert or mission-critical domains.

Interestingly, upon looking at “What kinds of procedures do you think these conversational agents [NOT] would be helpful for?”, we found that people believe that these agents should be capable of guiding people in technology related procedures (9/30 participants). Many people (16/30) interpreted procedures as capabilities of the agent, and listed virtual assistant capabilities. Interestingly, people mentioned cooking, household chores, and “simple procedures that are unchanging”. This may give some idea of why the ChatGPT agent was so effective in prior studies, as many of the procedures that were utilized were of this nature. When we looked at things that people do *not* think that these agents will work in, people mentioned that they may not be helpful for nuanced situations, complex tech issues, and complex problem-solving. They are also not recommended for critical (essential) tasks, tasks that require frequent updates or changes, subjective decision-making, emotional support, and physical tasks.

Upon looking at the rest of the usability questions, we see that there is an increase, although insignificant, in general usability. Further testing needs to be done to verify whether this increase is due to providing the correct procedure, or from the description of the conversational pattern that was given.

7.4.5 Overall findings

Summarizing our findings:

- Our results show that the inclusion of up-to-date, knowledge driven generated steps and facts, combined with descriptions of conversational patters, can provide more accurate and reliable procedures for users of conversational agents [H9]

- Our results show that LLM based agents are conversational, natural, and easy to use. [H10]

- Our results suggest that technical guidance with these conversational agents, especially in devices that have voice assistants, would be effective [H11]

- Our results suggest that for procedural guidance, LLM based conversational agents would perform well in general, unchanging procedures, but require some updating mechanism for domains that change yearly, and require even more factuality and reliability in mission-critical domains [H11]

- Our results suggest that voice assistants on devices would need to have procedural guidance as a complementary ability, and would need other functionality such as basic intents for weather, alarms, etc. [H11]

- Our results suggest that the inclusion of conversational pattern descriptions as prompts for the language models could potentially improve the overall usability, but more experimentation needs to be done on this [H11]

7.5 Future Guidelines

From our studies and our observations, even those listed in Appendices, B.4, B.5, we were able to gather the following insights to help inform future development of conversational agents for procedures.

- If developing an agent with an intent-based/slot-filling system, there needs to be a wide variety of examples of how people may phrase their intentions. Some examples of this phrasing are: "Can you help me with [X]", "How to/How do I do [X]", "I want to do [X]", "Show me how to do [X]", "[X]" (Where [X] is the task that guidance is being requested for)

- Similarly, for error handling, intent-based systems should handle multiple variations of requesting for help: "[X MISTAKE], How can I fix this?", "How can I fix [X]", "I did [X] and it should have been [Y]", "[X MISTAKE]"

- Having the functionality to give the steps one at a time is a nice to have, but possibly for shorter procedures around 1-3 steps, it may be convenient to give them all at once. For longer procedures, it may get tedious to keep saying continuers, so other forms of indicating continuation may want to be taken into account, such as a timer, or the ability to give all the steps at once to a user.
- To have the ability to restart, jump to a step with a summary of the context is also a nice to have, and some people want this behavior for error handling thinking that they may have done something wrong.
- If guidance is being performed in mission-critical domains, LLM-based systems should try to utilize retrieval methods or other forms of access to find up-to-date ways of performing procedures
- LLM models sometimes hallucinate steps if prompted on too specific a domain
- Many participants agree that these conversational agents are better suited for "simple" procedures, these would be procedures that may not require specialized knowledge and are less prone to user error
- Some participants are skeptical that these agents can be used for very specific/expert-level domains
- Step-by-step guidance is effective, but steps should be simplified as much as possible to ease a user's understanding

7.6 Limitations in Studies

7.6.1 Reading Instructions from a Web-page

Noticeably, in our work, we do not test the condition in which users are told to explicitly read the instructions from a website to complete a procedure. The primary reason for this is because that would involve no form of conversational interaction. Additionally, as was found in [44], people find reading instructions cognitively demanding. We leave it as future work to see how the voice guidance agents fare, at least in task completion, against having the user look up or read a set of instructions on how to perform a procedure.

7.6.2 Lack of usage tutorial

All the studies that were run would have benefitted from having a tutorial, prior to beginning the interactions. This would have helped users identify ways of how to formulate their queries, and may have decreased the intent/slot filling issues. We note this for future works that are done with intent-based agents.

7.7 Summary

In this seventh Chapter, in two separate studies, we evaluated the effectiveness of conversational agents in guiding users through procedures, including error recovery situations. In the first study, our findings showed that all three agents were capable of guiding most people through procedures, but our two intent-based agents need improvements in usability, particularly in the intent-matching and slot-filling capabilities. We also found that LLMs are very capable of providing guidance in most procedures, although their information may be out-of-date.

In the second study, we evaluated the agents' ability to provide guidance in error recovery situations. We found that all three agents were capable of guiding people in such situations, but the intent-based agent needed improvement in understanding capabilities. Our evidence suggested that maintaining context in these error interactions was essential, and step-by-step instructions, especially if clear and concise, could help guide people in procedures and out of errors. We also found that the LLM-based agent was much more flexible in error handling and could easily provide guidance in an exception process.

In the third and final study, we try to leverage the flexibility of the LLM agent in regard to intent recognition, slot-filling, and chit-chat capabilities and combine this with our knowledge driven step generation system and the contextual commonsense inference facts to build a hybrid system that is capable of identifying requests for procedural guidance and retrieving the top ranked facts and generated steps for it. The system uses this as a prompt to give guidance to a user. We find that this hybrid system is capable of providing more accurate and reliable knowledge than a baseline LLM. Additionally, we see that this hybrid agent that also utilizes descriptions of conversational patterns (as we will see in the next chapter) seems to improve usability of the system.

With all of this, we show that our contextual commonsense knowledge-based pipeline can be applied to end users effectively both in the form of an intent-based system and as a more capably hybrid knowledge-LLM to help guide them through procedures and for the recovery of procedures.

Chapter 8

Limitations and Discussion

The future depends on what you do today. -Mahatma Gandhi

8.1 Expert-Domain Guidance

While LLMs are excellent for general procedural guidance, their effectiveness diminishes in domains that are not well documented or seen multiple times in their training data. We examine the domain of video games for these agents. In particular, we examine getting an undead horse called Shadowmere in the Elder Scrolls V: Skyrim[1]. Skyrim has been a very popular video game since its release in 2011; it is a role-playing video game in which a player plays as a character that goes around a large game world performing quests.

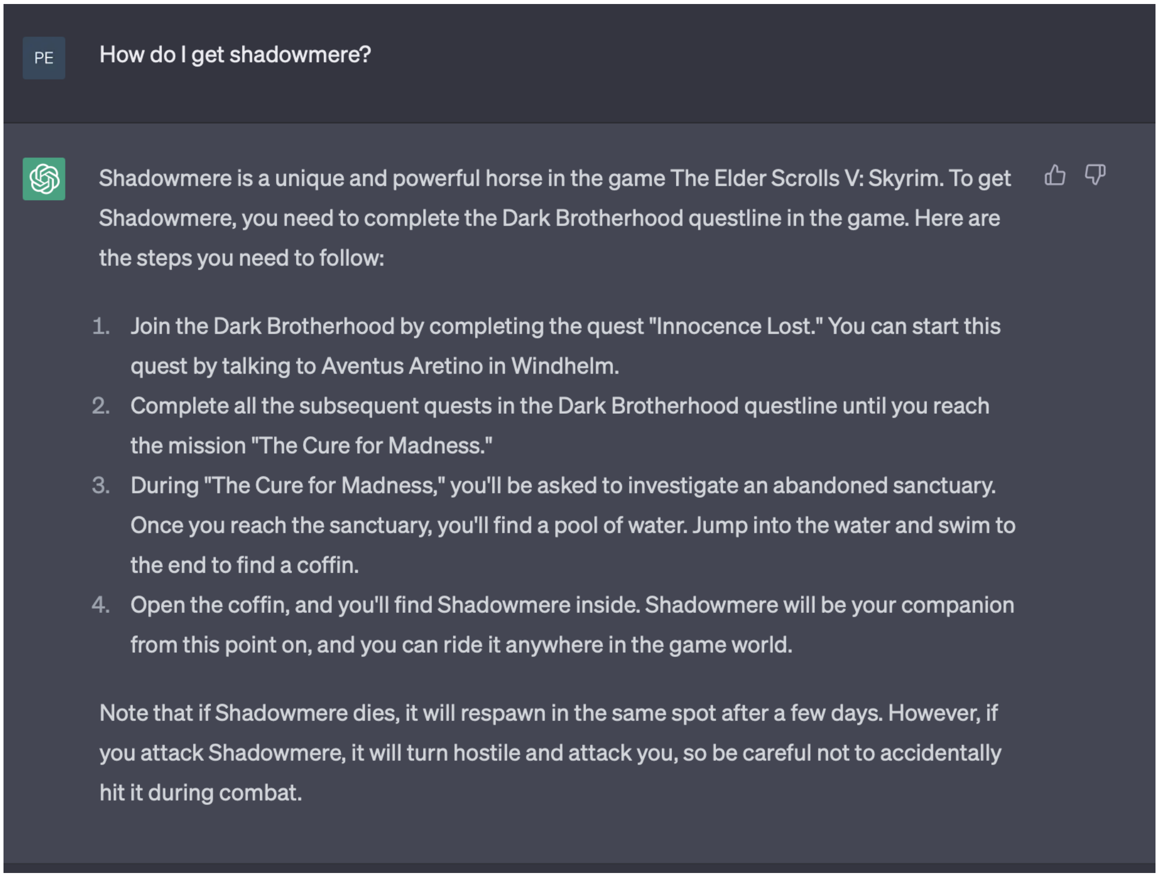
To get the Shadowmere (a magical horse in the video game), the instructions are the following[2]:

1. Start the quest: “The Cure for Madness”
2. Talk to Astrid
3. Find and Search Cicero’s room
4. Read Cicero’s diary

---

[1] `https://en.wikipedia.org/wiki/The_Elder_Scrolls_V:_Skyrim`

[2] Taken from: `https://elderscrolls.fandom.com/wiki/The_Cure_for_Madness`



How do I get shadowmere?

Shadowmere is a unique and powerful horse in the game The Elder Scrolls V: Skyrim. To get Shadowmere, you need to complete the Dark Brotherhood questline in the game. Here are the steps you need to follow:

1. Join the Dark Brotherhood by completing the quest "Innocence Lost." You can start this quest by talking to Aventus Aretino in Windhelm.
2. Complete all the subsequent quests in the Dark Brotherhood questline until you reach the mission "The Cure for Madness."
3. During "The Cure for Madness," you'll be asked to investigate an abandoned sanctuary. Once you reach the sanctuary, you'll find a pool of water. Jump into the water and swim to the end to find a coffin.
4. Open the coffin, and you'll find Shadowmere inside. Shadowmere will be your companion from this point on, and you can ride it anywhere in the game world.

Note that if Shadowmere dies, it will respawn in the same spot after a few days. However, if you attack Shadowmere, it will turn hostile and attack you, so be careful not to accidentally hit it during combat.

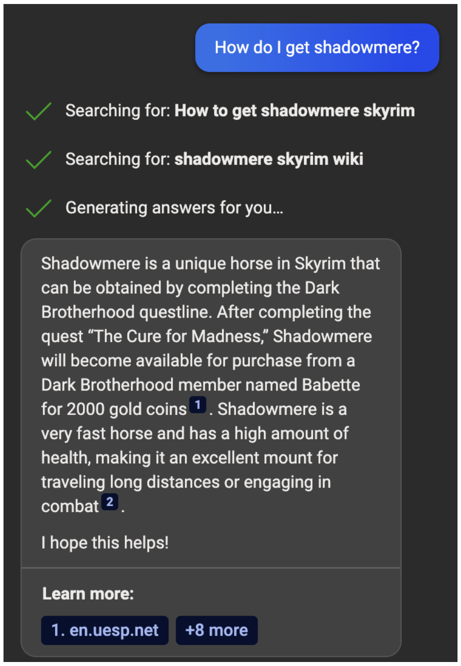
Figure 8-1: LLMs step generation performance in specific domain. People that have played the video game, will know that Shadowmere is a horse, and that horses are not found in a chest.

5. Talk to Astrid

6. Get Shadowmere

While it is a straightforward process, just consisting of completing a quest, when a LLM is prompted on this domain, we get a response such as the one in Figure 8-1. If we gave this objective to our KDPSG system, which relies on explicit knowledge to generate procedural steps, we would get the following steps:

- Step 1: Start the Cure for Madness quest.
- Step 2: Look for clues about where Cicero is.
- Step 3: Report Astrid.
- Step 4: Go to Dawnstar Sanctuary.



How do I get shadowmere?

Searching for: **How to get shadowmere skyrim**

Searching for: **shadowmere skyrim wiki**

Generating answers for you…

Shadowmere is a unique horse in Skyrim that can be obtained by completing the Dark Brotherhood questline. After completing the quest "The Cure for Madness," Shadowmere will become available for purchase from a Dark Brotherhood member named Babette for 2000 gold coins [1]. Shadowmere is a very fast horse and has a high amount of health, making it an excellent mount for traveling long distances or engaging in combat [2].

I hope this helps!

Figure 8-2: Bing chat, powered by GPT-4, failing on providing specific instructions for how to get Shadowmere.

- Step 5: Complete the quest to find Shadowmere.

What this suggests, is that very specific domain guidance, may not be very effective with LLMs. However a system may be able to do this task if provided with the knowledge for it. By augmenting a LLM with a retrieval mechanism this shortcoming might be fixed. However, that even with explicit retrieval mechanisms, LLMs may not be able to do this task on specific domains, as they might ignore the additional information. An example of this is utilizing the Bing-Chat system which has the Bing search engine to help it retrieve results and is also based on GPT-4, the successor to the GPT-3 systems. We show how it fails in Figure 8-2.

Overall, retrieval methods seem a very promising approach, but ways into effectively incorporating their retrievals need to be further explored.

8.2 Reasoning Assumptions

The work that we developed makes the assumption that the reasoner system is handed a set of assertions belongs to a certain step. While this may be acceptable to develop and test our systems, it limits the capabilities of the system to dynamically come up with procedures. A more flexible system should be able to search through all the knowledge that it has acquired to help it come up with a procedural step. This is what we attempted to do by testing the RAG-token model. However, we found that it was not very effective. To make this system work, we may need to rework the creation of documents to feed the retriever. We initially made every document contain an assertion, it may be more effective for a document to contain clusters of similar assertions. When thinking about this problem, future endeavors are faced with a search at scale issue. We have a large knowledge base of contextual assertions, from which we need to retrieve and order a subset of them. We have seen in our work that it is indeed possible to operate in this knowledge space, adding information helps contextualize error handling. Future work should look at how to adapt the knowledge that the realizer has to personalization inquiries from a user (i.e., a user modifies a certain aspect of a procedure, and the rest of the procedure generation should take this into account.

8.3 Intent recognition systems

From the feedback and the results that we received from our user studies, it is evident that an intent-recognition system needs to be heavily improved, especially for open-domain procedural guidance. Much of the feedback that was received, and seen in the results of the usability, concerned the system "not understanding" what users wanted. This is due mostly to new phrasing of what a user wants to do (the intent matcher does not have enough examples to discern that it is a request for guidance/fixing an error). Because of this, we recommend that future work that wants to explore this area take into consideration utilizing more flexible intent recognition systems, or replacing the intent recognition system altogether with a knowledge-enhanced large language model.

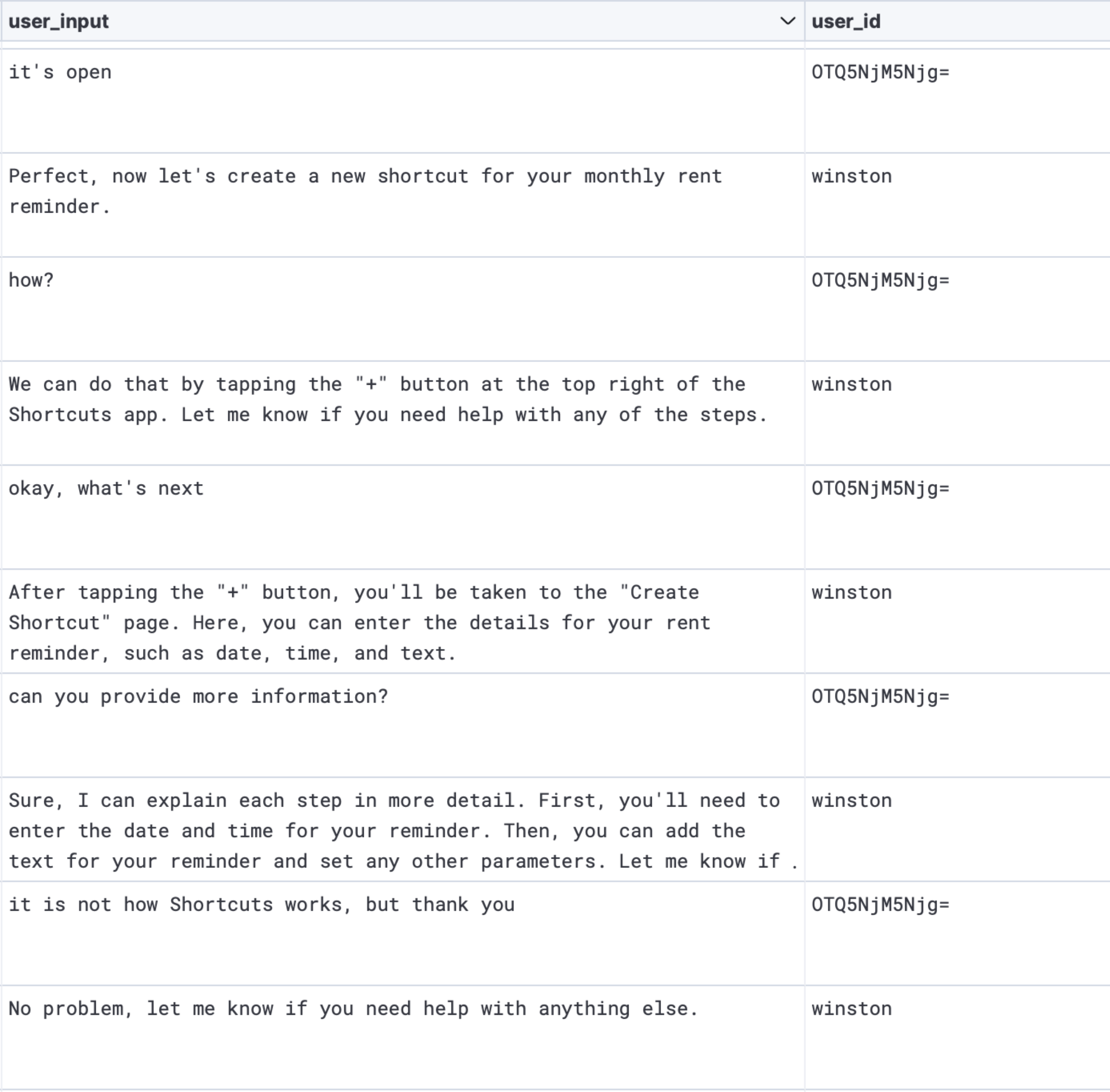
8.4 Large Language Models - Lack of conversational patterns

Language models are trained on large datasets of text, which can include conversational data, but also includes many other types of text such as news articles, books, and scientific papers. As a result, these models may not always follow conversational patterns when generating text. For example, they may generate responses that are too formal or too technical for a conversation, or they may generate responses that are not relevant to the conversation at hand. Additionally, language models may not always understand the context of a conversation or be able to generate responses that are appropriate for the situation. This has been called a lack of conversational intent [11]. One example of this can be seen in Figure 8-3, where the LLM-based agent just keeps responding without actually providing the guidance that was required. In many other cases, what happens is that the language model gives a response that is much longer than what the working memory of a person serves. This can be seen in Figure 8-4.

One potential way to address this issue, is to inform the language model of how a procedural interaction or extended telling works. We now try this with the procedure of "how to write a check" in two different ways: giving a summary paragraph of how extended telling works, and by giving the LLM the conversational pattern.

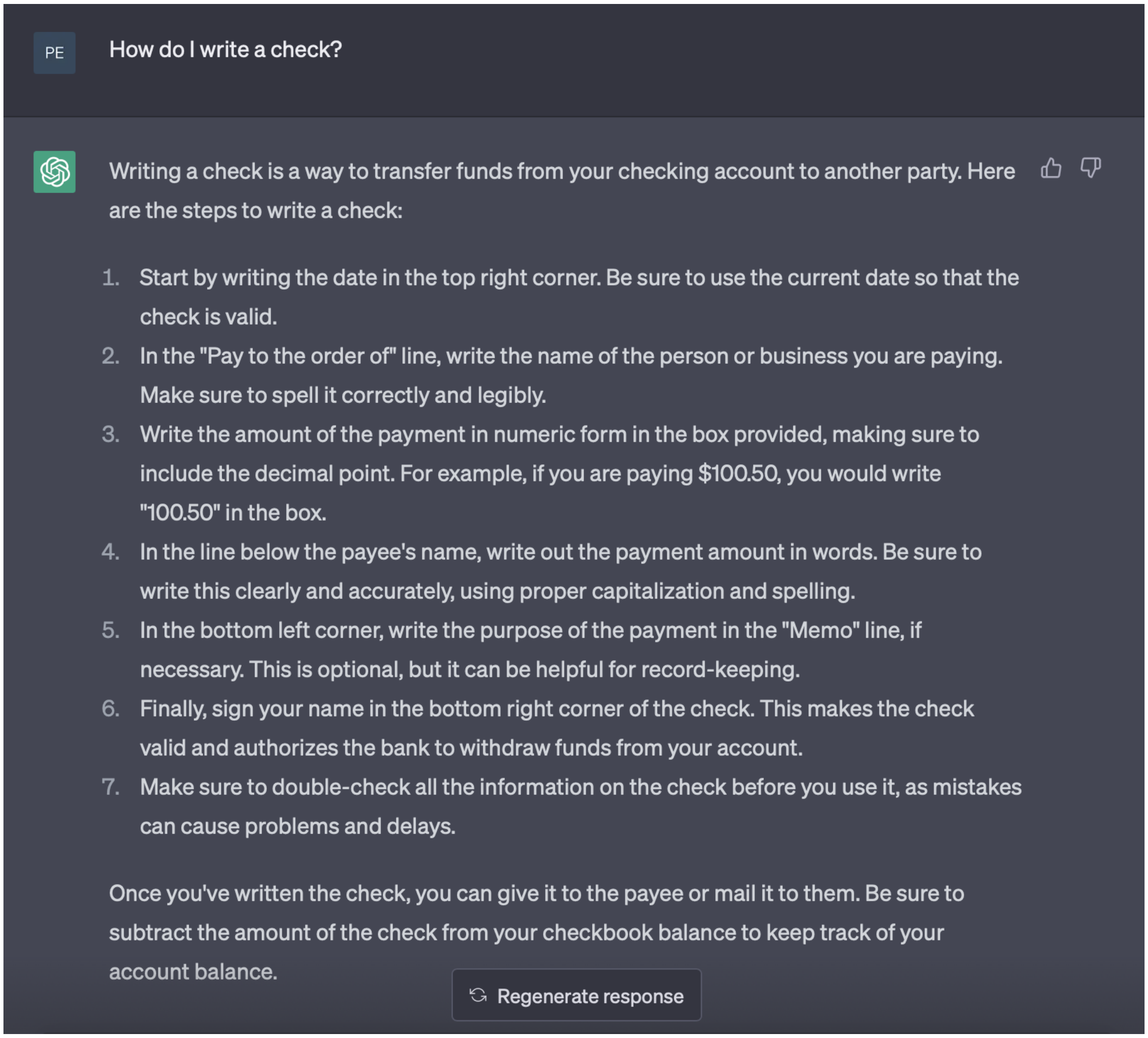
Firstly, we give the following directive to the LLM:

> *If asked for instructions on how to do something use the following pattern to convey the instructions. First confirm what the person wants to get guidance on. Then begin the interaction by providing one step at a time. You will wait for a person to confirm/acknowledge the step, and then you will provide the next one. If someone asks you a question about the procedure, you will try to answer it, but remind the user that you should continue. If you are giving instructions on a procedure, and a person makes a mistake that cannot be addressed with a question, you will guide them on how to fix the issue with the same strategy as above. When the*



user_input	user_id
it's open	OTQ5NjM5Njg=
Perfect, now let's create a new shortcut for your monthly rent reminder.	winston
how?	OTQ5NjM5Njg=
We can do that by tapping the "+" button at the top right of the Shortcuts app. Let me know if you need help with any of the steps.	winston
okay, what's next	OTQ5NjM5Njg=
After tapping the "+" button, you'll be taken to the "Create Shortcut" page. Here, you can enter the details for your rent reminder, such as date, time, and text.	winston
can you provide more information?	OTQ5NjM5Njg=
Sure, I can explain each step in more detail. First, you'll need to enter the date and time for your reminder. Then, you can add the text for your reminder and set any other parameters. Let me know if .	winston
it is not how Shortcuts works, but thank you	OTQ5NjM5Njg=
No problem, let me know if you need help with anything else.	winston

Figure 8-3: LLM-based conversation in which the language model does not begin an extended telling or does not move the conversation forward towards guidance.



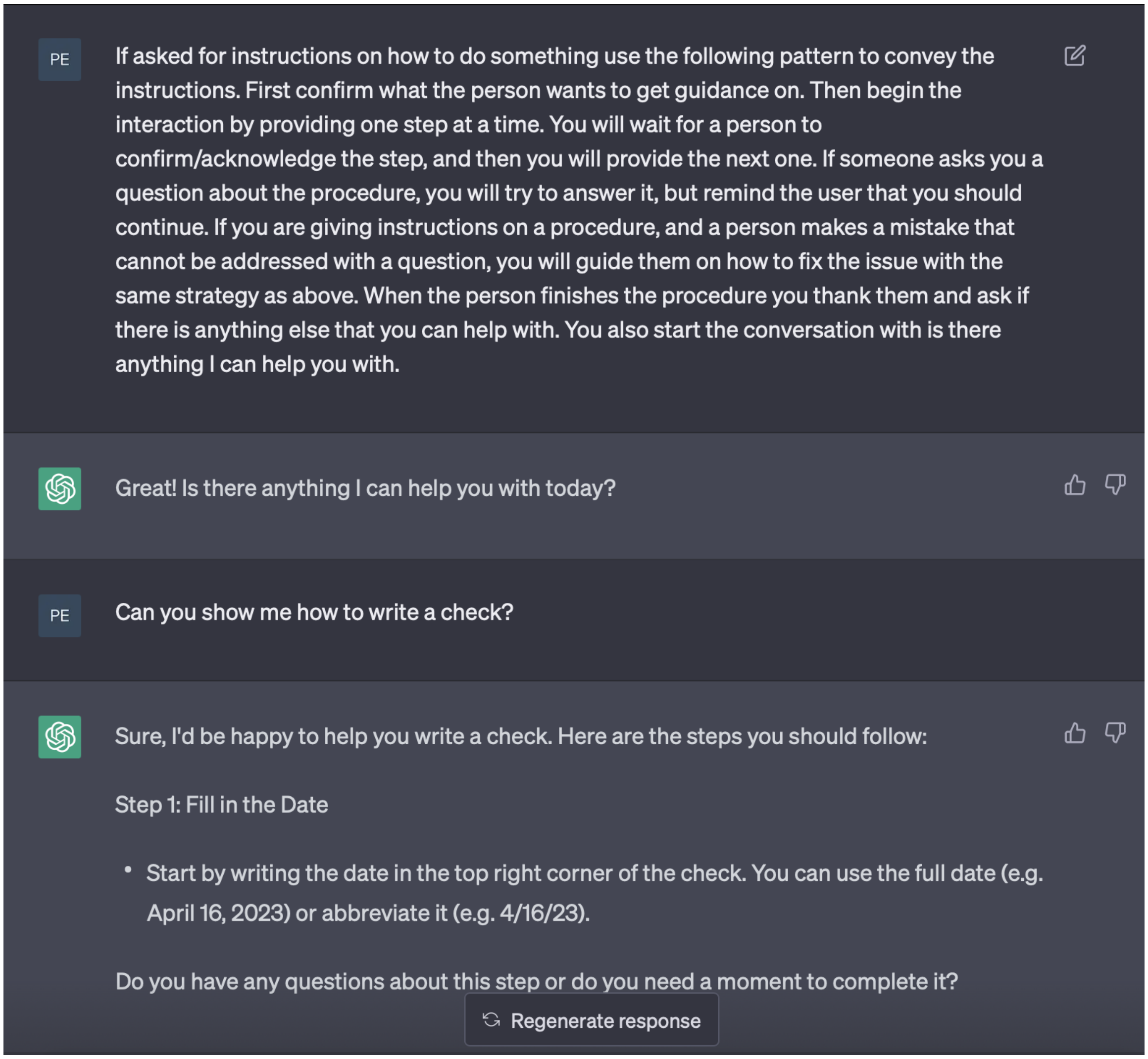
How do I write a check?

Writing a check is a way to transfer funds from your checking account to another party. Here are the steps to write a check:

1. Start by writing the date in the top right corner. Be sure to use the current date so that the check is valid.
2. In the "Pay to the order of" line, write the name of the person or business you are paying. Make sure to spell it correctly and legibly.
3. Write the amount of the payment in numeric form in the box provided, making sure to include the decimal point. For example, if you are paying $100.50, you would write "100.50" in the box.
4. In the line below the payee's name, write out the payment amount in words. Be sure to write this clearly and accurately, using proper capitalization and spelling.
5. In the bottom left corner, write the purpose of the payment in the "Memo" line, if necessary. This is optional, but it can be helpful for record-keeping.
6. Finally, sign your name in the bottom right corner of the check. This makes the check valid and authorizes the bank to withdraw funds from your account.
7. Make sure to double-check all the information on the check before you use it, as mistakes can cause problems and delays.

Once you've written the check, you can give it to the payee or mail it to them. Be sure to subtract the amount of the check from your checkbook balance to keep track of your account balance.

Figure 8-4: Conversation in which a LLM is not given directions on how to interact with users.


PE

If asked for instructions on how to do something use the following pattern to convey the instructions. First confirm what the person wants to get guidance on. Then begin the interaction by providing one step at a time. You will wait for a person to confirm/acknowledge the step, and then you will provide the next one. If someone asks you a question about the procedure, you will try to answer it, but remind the user that you should continue. If you are giving instructions on a procedure, and a person makes a mistake that cannot be addressed with a question, you will guide them on how to fix the issue with the same strategy as above. When the person finishes the procedure you thank them and ask if there is anything else that you can help with. You also start the conversation with is there anything I can help you with.

Great! Is there anything I can help you with today?

PE

Can you show me how to write a check?

Sure, I'd be happy to help you write a check. Here are the steps you should follow:

Step 1: Fill in the Date

- Start by writing the date in the top right corner of the check. You can use the full date (e.g. April 16, 2023) or abbreviate it (e.g. 4/16/23).

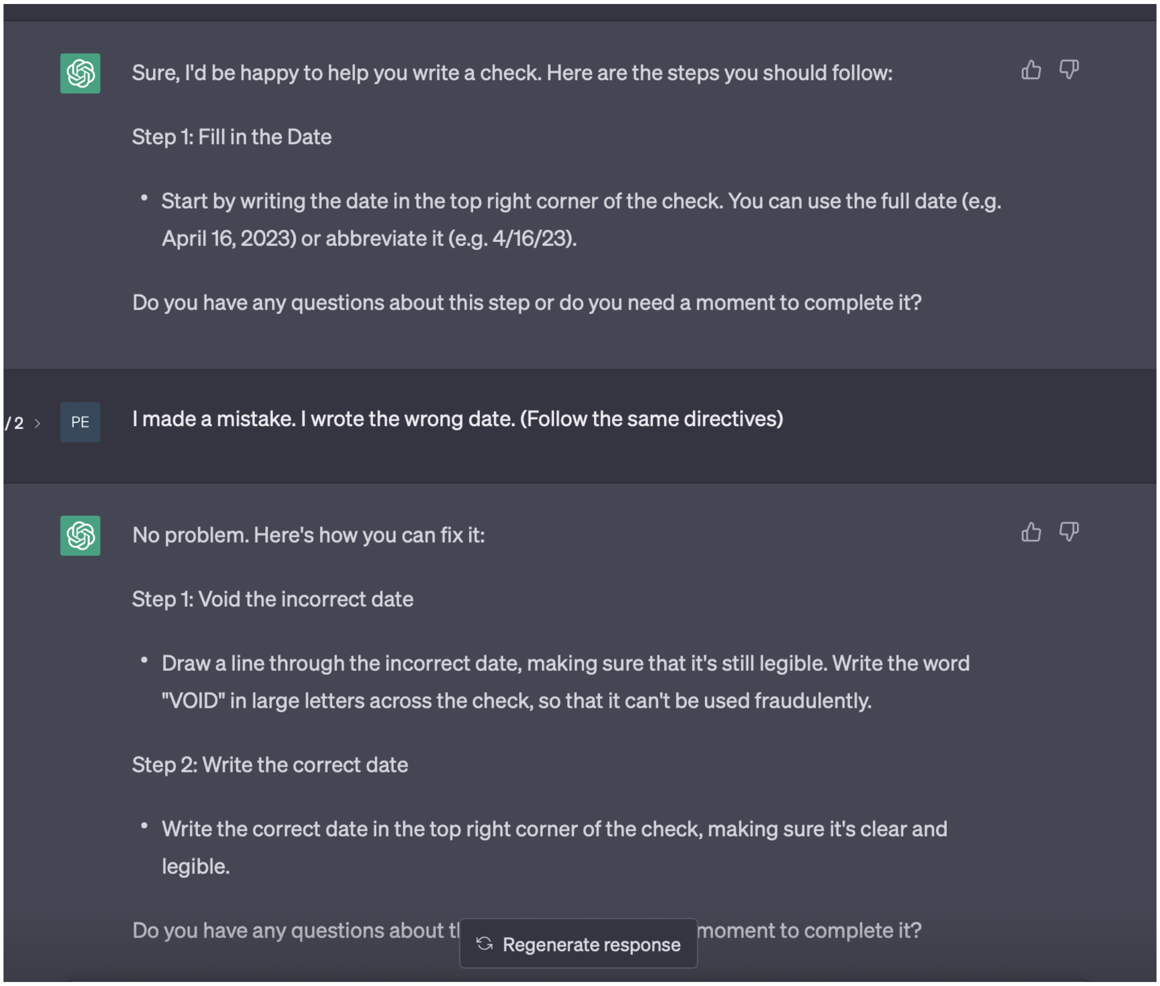
Do you have any questions about this step or do you need a moment to complete it?

Figure 8-5: Conversation in which a LLM is given directions on how to interact with users. We note that the answers are shorter and to the point, and wait for a user to continue.

> *person finishes the procedure you thank them and ask if there is anything else that you can help with. You also start the conversation with is there anything I can help you with.*

We observe some immediate effects in Figure 8-5 and Figure 8-6

The model tries to structure its responses and wait for a user before continuing. However, it does not follow the description perfectly, and eventually gives two steps in one. This avenue of research seems very promising, especially because while these agents are “conversational” they are conversational in a visual-centric [108] manner



Sure, I'd be happy to help you write a check. Here are the steps you should follow:

Step 1: Fill in the Date

- Start by writing the date in the top right corner of the check. You can use the full date (e.g. April 16, 2023) or abbreviate it (e.g. 4/16/23).

Do you have any questions about this step or do you need a moment to complete it?

/2 PE I made a mistake. I wrote the wrong date. (Follow the same directives)

No problem. Here's how you can fix it:

Step 1: Void the incorrect date

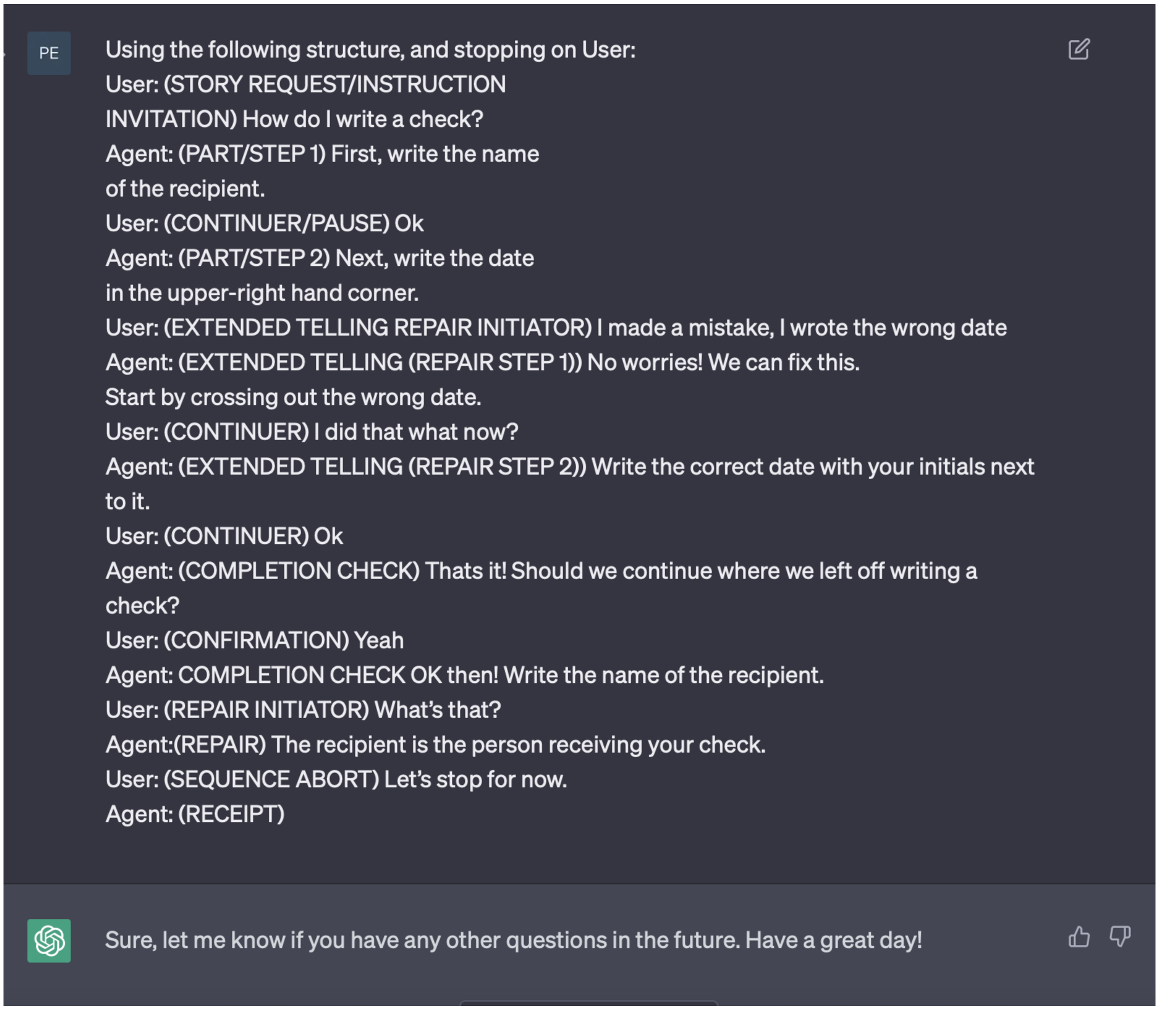
- Draw a line through the incorrect date, making sure that it's still legible. Write the word "VOID" in large letters across the check, so that it can't be used fraudulently.

Step 2: Write the correct date

- Write the correct date in the top right corner of the check, making sure it's clear and legible.

Do you have any questions about tl moment to complete it?

Figure 8-6: Conversation in which a LLM is given directions on how to interact with users. We note that the model does not exactly follow the given instructions, when an error occurs, but it approximates them.



Using the following structure, and stopping on User:
User: (STORY REQUEST/INSTRUCTION
INVITATION) How do I write a check?
Agent: (PART/STEP 1) First, write the name
of the recipient.
User: (CONTINUER/PAUSE) Ok
Agent: (PART/STEP 2) Next, write the date
in the upper-right hand corner.
User: (EXTENDED TELLING REPAIR INITIATOR) I made a mistake, I wrote the wrong date
Agent: (EXTENDED TELLING (REPAIR STEP 1)) No worries! We can fix this.
Start by crossing out the wrong date.
User: (CONTINUER) I did that what now?
Agent: (EXTENDED TELLING (REPAIR STEP 2)) Write the correct date with your initials next
to it.
User: (CONTINUER) Ok
Agent: (COMPLETION CHECK) Thats it! Should we continue where we left off writing a
check?
User: (CONFIRMATION) Yeah
Agent: COMPLETION CHECK OK then! Write the name of the recipient.
User: (REPAIR INITIATOR) What's that?
Agent:(REPAIR) The recipient is the person receiving your check.
User: (SEQUENCE ABORT) Let's stop for now.
Agent: (RECEIPT)

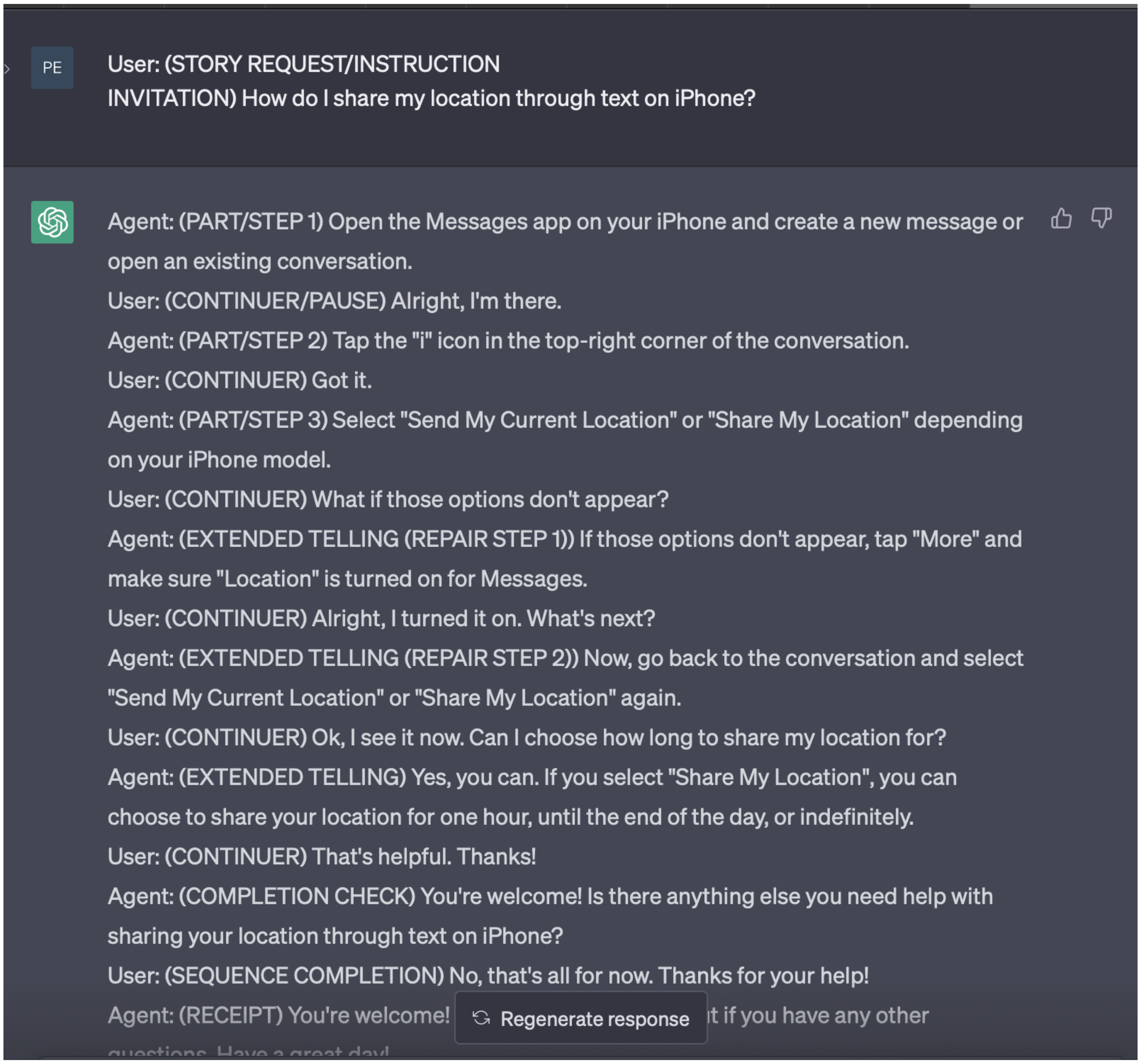
Sure, let me know if you have any other questions in the future. Have a great day!

Figure 8-7: Providing an example conversation as a prompt for a LLM.

(i.e., you would probably need to read the responses that they give to understand them rather than just simply hearing them out loud), and there has not been research on how to enforce conversation-centric responses.

We also tried to give the system an example extended telling conversation with annotations, and see if it followed the pattern. However, the system hallucinated a whole conversation (albeit somewhat correctly) on how the interaction would be. These can be seen in Figures 8-7 and 8-8

We see some success in this, however the generation process needs to be controlled further, and examples of different turn types/sequences need to be given. Altogether, there needs to be future work in incorporating well known conversational patterns



User: (STORY REQUEST/INSTRUCTION INVITATION) How do I share my location through text on iPhone?

Agent: (PART/STEP 1) Open the Messages app on your iPhone and create a new message or open an existing conversation.
User: (CONTINUER/PAUSE) Alright, I'm there.
Agent: (PART/STEP 2) Tap the "i" icon in the top-right corner of the conversation.
User: (CONTINUER) Got it.
Agent: (PART/STEP 3) Select "Send My Current Location" or "Share My Location" depending on your iPhone model.
User: (CONTINUER) What if those options don't appear?
Agent: (EXTENDED TELLING (REPAIR STEP 1)) If those options don't appear, tap "More" and make sure "Location" is turned on for Messages.
User: (CONTINUER) Alright, I turned it on. What's next?
Agent: (EXTENDED TELLING (REPAIR STEP 2)) Now, go back to the conversation and select "Send My Current Location" or "Share My Location" again.
User: (CONTINUER) Ok, I see it now. Can I choose how long to share my location for?
Agent: (EXTENDED TELLING) Yes, you can. If you select "Share My Location", you can choose to share your location for one hour, until the end of the day, or indefinitely.
User: (CONTINUER) That's helpful. Thanks!
Agent: (COMPLETION CHECK) You're welcome! Is there anything else you need help with sharing your location through text on iPhone?
User: (SEQUENCE COMPLETION) No, that's all for now. Thanks for your help!
Agent: (RECEIPT) You're welcome! t if you have any other questions. Have a great day!

Figure 8-8: Follow up of providing an example conversation as a prompt for a LLM. We see that the LLM hallucinates an entire conversation to a certain correct degree.

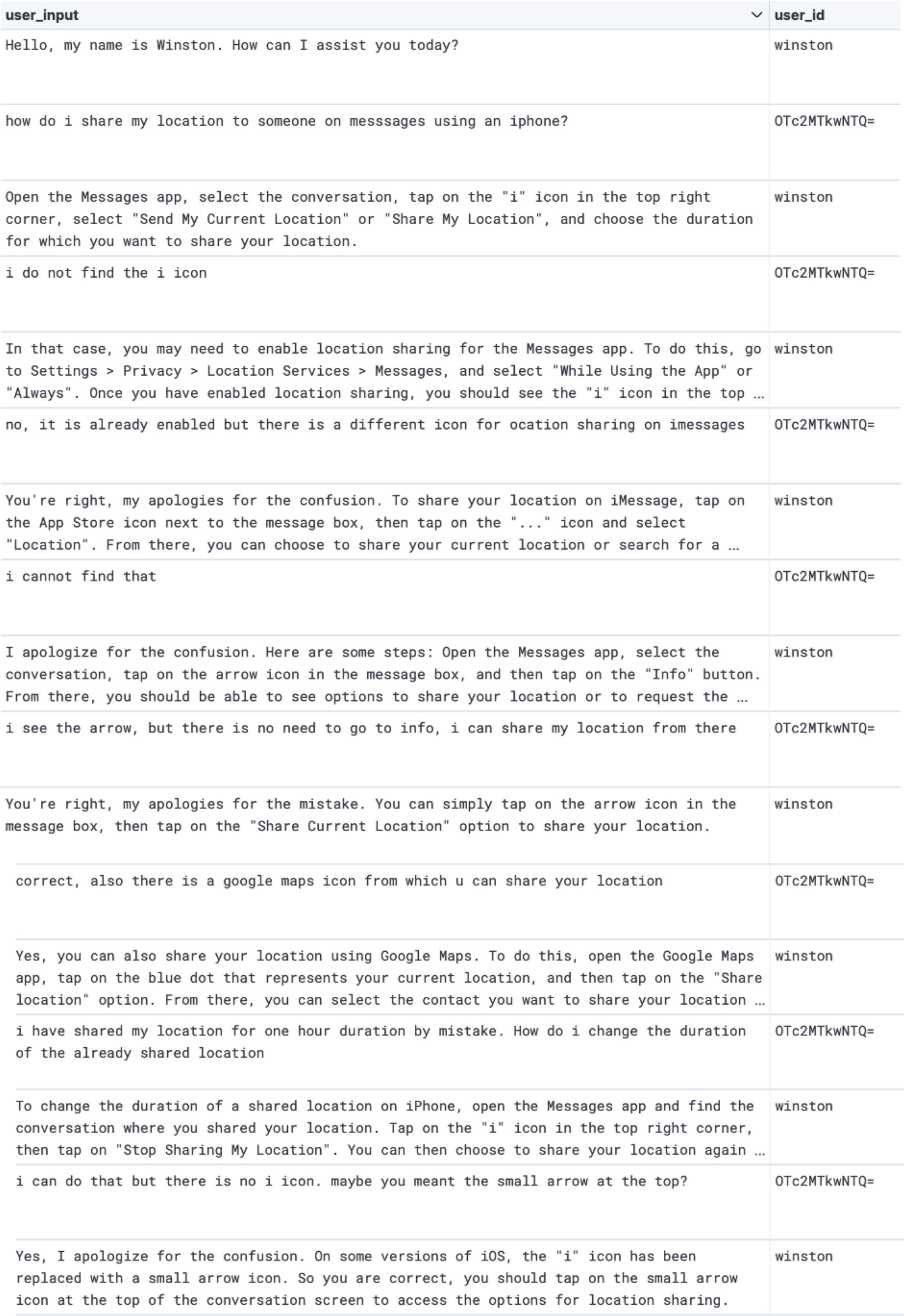
into these LLMs, especially if they are to be used for voice/chat guidance.

8.4.1 Outdated training data

From the feedback and observations that we gathered, although LLMs are good for procedural guidance, they are limited to what they know based on their training data, and are limited in the generality of the procedures that they know. What this implies is that there needs to be some way to inform these models of knowledge that they may not have. Retrieval methods are an excellent alternative for this, but these methods typically consist of finding an exact document with the results. However, as we have argued before, commonsense knowledge may be needed in a wide variety of tasks that a user might engage in with an agent. We believe that one possibly useful alternative to explicit retrieval, is to supply organized/planned assertions for models so that it can use them as they see fit. In our particular case, we could supply a LLM with the contextual commonsense assertions that our contextual commonsense inference model generates about a relevant text to a model to guide its language generation. We now take an example from one of our studies. In this particular interaction, without any additional knowledge, the language model gives erroneous information of what needs to be done, because its information is outdated. This can be seen in Figure 8-9.

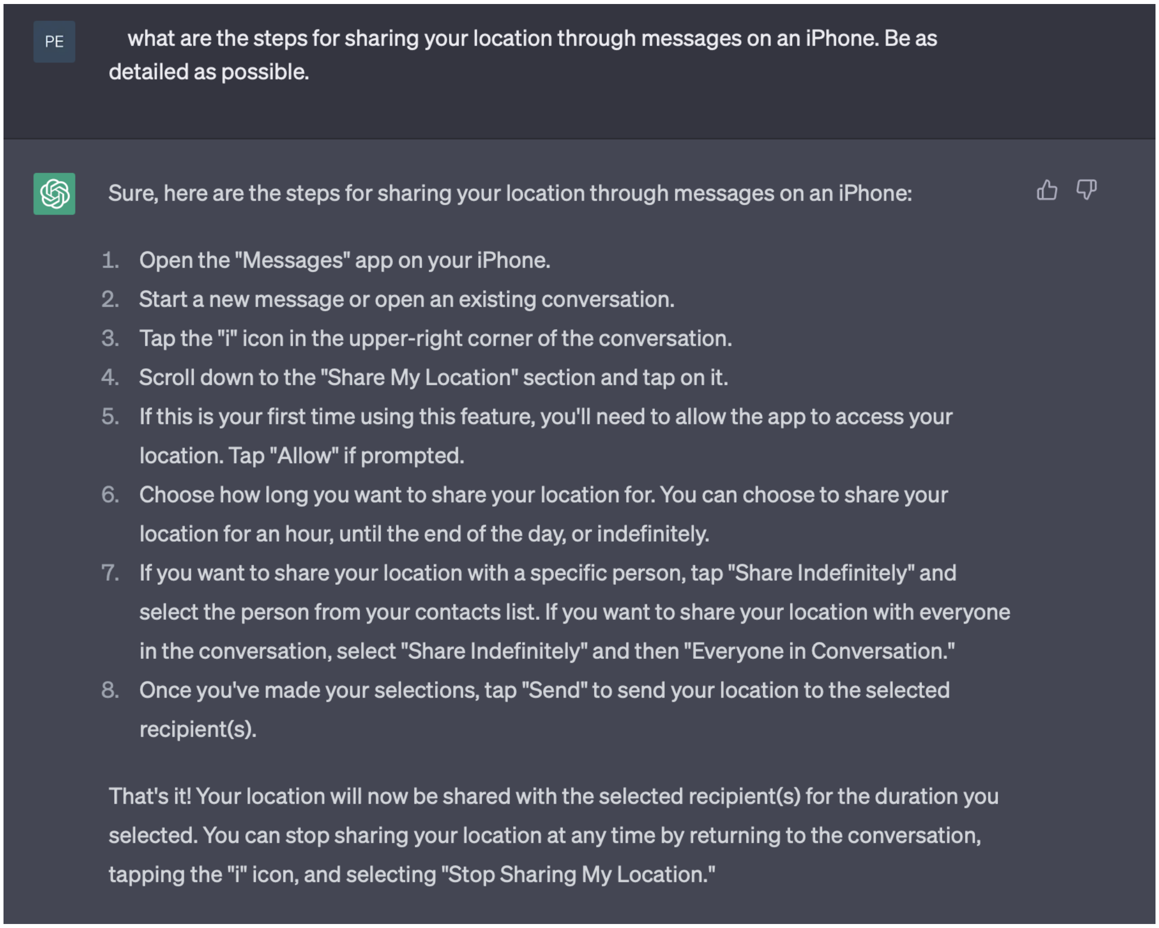
We see in this interaction that first the agent tries a general sub-procedure to try and fix the problem (give location permissions to the app), then it moves on to use information from other procedures (sharing location by downloading some app from the Messages App store). We see much later on in the interaction that even though the model was corrected on what the options were, it reverts to what it knows. From this interaction, we see that the LLM's understanding of what can be done in the Messages app is limited and outdated, and when pressed on to give more information, the agent simply hallucinates some other very general procedure.

If we look at what the language model knows in a zero-shot setting just for this procedure, we see the following in Figure 8-10. Now, if we prompt it with the follow-



user_input	user_id
Hello, my name is Winston. How can I assist you today?	winston
how do i share my location to someone on messsages using an iphone?	OTc2MTkwNTQ=
Open the Messages app, select the conversation, tap on the "i" icon in the top right corner, select "Send My Current Location" or "Share My Location", and choose the duration for which you want to share your location.	winston
i do not find the i icon	OTc2MTkwNTQ=
In that case, you may need to enable location sharing for the Messages app. To do this, go to Settings > Privacy > Location Services > Messages, and select "While Using the App" or "Always". Once you have enabled location sharing, you should see the "i" icon in the top …	winston
no, it is already enabled but there is a different icon for ocation sharing on imessages	OTc2MTkwNTQ=
You're right, my apologies for the confusion. To share your location on iMessage, tap on the App Store icon next to the message box, then tap on the "..." icon and select "Location". From there, you can choose to share your current location or search for a …	winston
i cannot find that	OTc2MTkwNTQ=
I apologize for the confusion. Here are some steps: Open the Messages app, select the conversation, tap on the arrow icon in the message box, and then tap on the "Info" button. From there, you should be able to see options to share your location or to request the …	winston
i see the arrow, but there is no need to go to info, i can share my location from there	OTc2MTkwNTQ=
You're right, my apologies for the mistake. You can simply tap on the arrow icon in the message box, then tap on the "Share Current Location" option to share your location.	winston
correct, also there is a google maps icon from which u can share your location	OTc2MTkwNTQ=
Yes, you can also share your location using Google Maps. To do this, open the Google Maps app, tap on the blue dot that represents your current location, and then tap on the "Share location" option. From there, you can select the contact you want to share your location …	winston
i have shared my location for one hour duration by mistake. How do i change the duration of the already shared location	OTc2MTkwNTQ=
To change the duration of a shared location on iPhone, open the Messages app and find the conversation where you shared your location. Tap on the "i" icon in the top right corner, then tap on "Stop Sharing My Location". You can then choose to share your location again …	winston
i can do that but there is no i icon. maybe you meant the small arrow at the top?	OTc2MTkwNTQ=
Yes, I apologize for the confusion. On some versions of iOS, the "i" icon has been replaced with a small arrow icon. So you are correct, you should tap on the small arrow icon at the top of the conversation screen to access the options for location sharing.	winston

Figure 8-9: Conversation in which a LLM does not have up-to-date information and begins hallucinating when pressed on a topic that it does not have knowledge on.



what are the steps for sharing your location through messages on an iPhone. Be as detailed as possible.

Sure, here are the steps for sharing your location through messages on an iPhone:

1. Open the "Messages" app on your iPhone.
2. Start a new message or open an existing conversation.
3. Tap the "i" icon in the upper-right corner of the conversation.
4. Scroll down to the "Share My Location" section and tap on it.
5. If this is your first time using this feature, you'll need to allow the app to access your location. Tap "Allow" if prompted.
6. Choose how long you want to share your location for. You can choose to share your location for an hour, until the end of the day, or indefinitely.
7. If you want to share your location with a specific person, tap "Share Indefinitely" and select the person from your contacts list. If you want to share your location with everyone in the conversation, select "Share Indefinitely" and then "Everyone in Conversation."
8. Once you've made your selections, tap "Send" to send your location to the selected recipient(s).

That's it! Your location will now be shared with the selected recipient(s) for the duration you selected. You can stop sharing your location at any time by returning to the conversation, tapping the "i" icon, and selecting "Stop Sharing My Location."

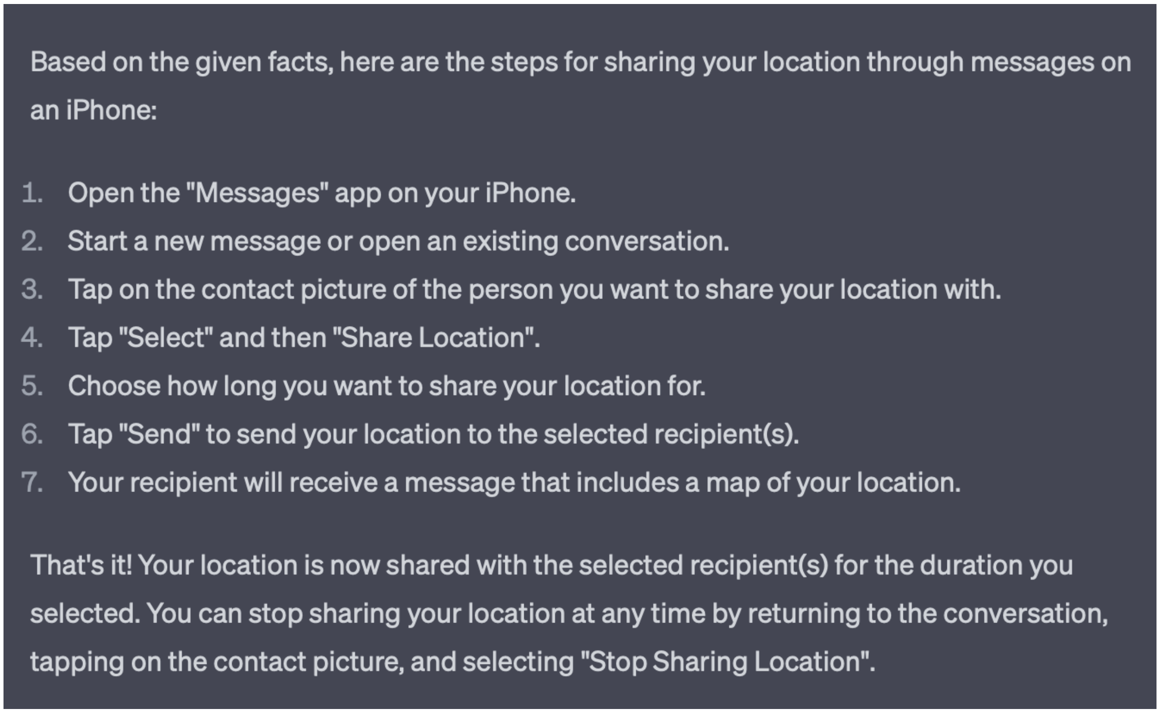
Figure 8-10: Outdated, zero-shot, ChatGPT generated procedure for how to share location on an iPhone.

ing:

> ***Only using the following facts:*** *Your iPhone desires share location. Open can be characterized by being/having text messaging. The messages app used for send a message. Your iPhone used for send a message. The person located or found at/in/on contact. Your location to do this, one requires contact. Want includes the event/action share location. Select because share location. Would want is a step towards accomplishing the goal share location. Select can receive or be affected by the action the location. Select desires share location. The contact picture is a location. The contact picture is related to location. The contact picture has, possesses or contains location. Tap has, possesses or contains location. Tap has the subevent share location. The contact picture is a form of the contact message. The top is a form of the top. The contact picture used for sharing location. "Select is related to location. ", "Send made (up) of location. ", "Choose is related to location. ", "Will see is a form of see. ", "Will see is related to location. ", "Your recipient is related to location. ", "Send is related to location. ", "The map made (up) of location. ", "See is related to location. ", "Share is defined as location. ", "Want is related to location. ", "Select send is related to location. ", "Share made (up) of location. ", "Your recipient is/are capable of location. ", "Share is related to location. ",* ***Based only on the list of facts, what are the steps for sharing your location through messages on an iPhone. Be as detailed as possible.***

As seen in Figure 8-11, we see that the language model is actually able to understand the facts and piece them together in such a way as to generate a more correct/up-to-date procedure.

Now, if we had some background process to perform contextual commonsense inference, and supply a list of essential steps, about a process, we could utilize this to keep a language model that has old information up-to-date on how to do modern



Based on the given facts, here are the steps for sharing your location through messages on an iPhone:

1. Open the "Messages" app on your iPhone.
2. Start a new message or open an existing conversation.
3. Tap on the contact picture of the person you want to share your location with.
4. Tap "Select" and then "Share Location".
5. Choose how long you want to share your location for.
6. Tap "Send" to send your location to the selected recipient(s).
7. Your recipient will receive a message that includes a map of your location.

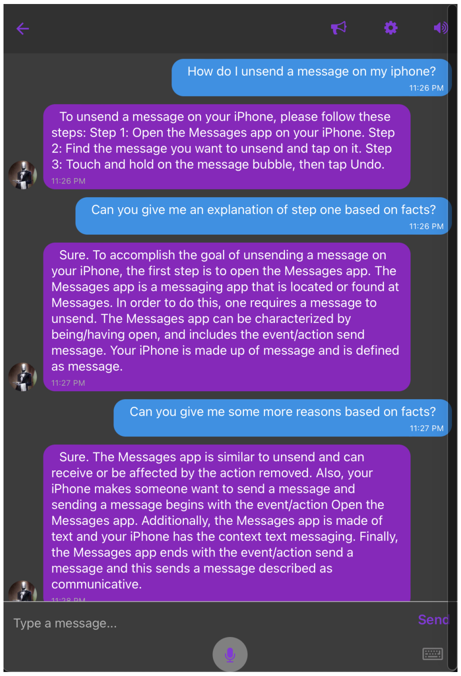
That's it! Your location is now shared with the selected recipient(s) for the duration you selected. You can stop sharing your location at any time by returning to the conversation, tapping on the contact picture, and selecting "Stop Sharing Location".

Figure 8-11: Generated text in which a LLM is given contextual commonsense assertions to help with procedural step generation.

processes. Another alternative is to retrieve a web page with the actual steps. In contrast to a simple retrieval and augmentation, some benefits that adding contextual assertions brings is that it makes very explicit the facts, both domain specific and commonsense that govern a procedure, thereby making constraints explicit for a language model. This exploration was what we utilized to develop our Hybrid model.

8.5 Explainability

One reason that we followed the route of generating and incorporating a knowledge graph, is because it can provide a visualizable explanation of what makes up a step (both explicitly and implicitly). Before the advent of very large language models, we needed to find a way to supply information that the model may not know (e.g., commonsense information). Now, however, very large language models have the capability of storing a vast amount of both factual and contextual knowledge[178]. One benefit, in this age of the LLM, of including a knowledge graph in our systems, is that it can give the user some insight into what the model used to reason to generate the steps that it was giving.



How do I unsend a message on my iphone?
11:26 PM

To unsend a message on your iPhone, please follow these steps: Step 1: Open the Messages app on your iPhone. Step 2: Find the message you want to unsend and tap on it. Step 3: Touch and hold on the message bubble, then tap Undo.
11:26 PM

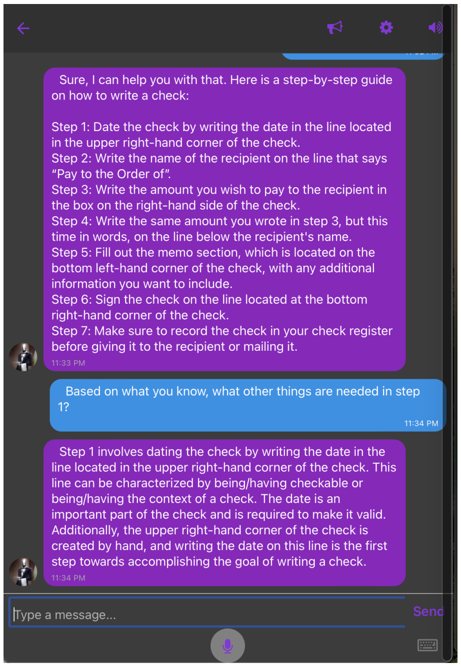
Can you give me an explanation of step one based on facts?
11:26 PM

Sure. To accomplish the goal of unsending a message on your iPhone, the first step is to open the Messages app. The Messages app is a messaging app that is located or found at Messages. In order to do this, one requires a message to unsend. The Messages app can be characterized by being/having open, and includes the event/action send message. Your iPhone is made up of message and is defined as message.
11:27 PM

Can you give me some more reasons based on facts?
11:27 PM

Sure. The Messages app is similar to unsend and can receive or be affected by the action removed. Also, your iPhone makes someone want to send a message and sending a message begins with the event/action Open the Messages app. Additionally, the Messages app is made of text and your iPhone has the context text messaging. Finally, the Messages app ends with the event/action send a message and this sends a message described as communicative.

Figure 8-12: Asking the hybrid KDPSG + LLM model to explain its reasoning for a specific step that the base model does not know how to do.



Sure, I can help you with that. Here is a step-by-step guide on how to write a check:

Step 1: Date the check by writing the date in the line located in the upper right-hand corner of the check.
Step 2: Write the name of the recipient on the line that says "Pay to the Order of".
Step 3: Write the amount you wish to pay to the recipient in the box on the right-hand side of the check.
Step 4: Write the same amount you wrote in step 3, but this time in words, on the line below the recipient's name.
Step 5: Fill out the memo section, which is located on the bottom left-hand corner of the check, with any additional information you want to include.
Step 6: Sign the check on the line located at the bottom right-hand corner of the check.
Step 7: Make sure to record the check in your check register before giving it to the recipient or mailing it.
11:33 PM

Based on what you know, what other things are needed in step 1?
11:34 PM

Step 1 involves dating the check by writing the date in the line located in the upper right-hand corner of the check. This line can be characterized by being/having checkable or being/having the context of a check. The date is an important part of the check and is required to make it valid. Additionally, the upper right-hand corner of the check is created by hand, and writing the date on this line is the first step towards accomplishing the goal of writing a check.
11:34 PM

Figure 8-13: Asking the hybrid KDPSG + LLM model for an explanation of a known step based on additional KG information

In Figures 8-12 and 8-13, we ask our Hybrid KDPSG+LLM agent to give an explanation on a certain step based on what it knows. Figure 8-12 asks it to explain a step that the base LLM model has outdated/unseen information on. We see that from the knowledge graph that we give it (based on the spelling that the agent uses), that it can begin to pick out facts that may be essential to understand the steps from the knowledge we supplied. We also see that this is applicable to seen procedures also. Although this is an exploration into this area, our Hybrid LLM+KDPSG system is capable of using the knowledge graph that was given in a way that we could post process and visualize to understand its reasoning.

8.6 Summary

In this eight Chapter, we discussed some of the limitations found in our work. In particular, the limitations of large language models (LLMs) in providing procedural guidance for specific domains. While LLMs excel in providing general procedural guidance, their effectiveness decreases in domains that are not well-documented or not seen multiple times in their training data. We also note their lack of rigid utilization of conversational patterns, which can result in generating responses that are useless for advancing a conversational. We suggest a promising direction of informing the language model of how a procedural interaction or extended telling works to address this issue. We also look at a promising direction of utilizing our contextual commonsense inference system for informing LLMs of the necessary information to do a procedure. We also explore the challenges in creating a flexible intent-based system such as the one we use for general procedure handling, and how this may be improved by leveraging LLMs to serve as the intent-recognition and interaction system. We looked at limitations in our reasoner system, and how a more general knowledge retrieval system may improve the system's ability to generate procedures on the fly.

Chapter 9

Contributions and Future Work

Don't cry because it's over, smile because it happened.-Dr. Seuss

9.1 Future Work

9.1.1 Embodiment

Although the system that we have mentioned up until now has been focused on being purely conversational with no other interface, it is possible to place it inside an embodied conversational agent (ECA). This additional embodiment may provide more focus and attention from users of the system. Embodiment may be beneficial because an embodied conversation agent will elicit significantly more sympathetic social behavior than those agents without physical form. This frames embodiment as a means to natural interaction.[96] Embodied conversational agents are "specifically conversational in their behaviors, and specifically human-like in the way they use their bodies in conversation". [96] In addition to this, if the embodied system has a screen, we can leverage it to provide visualizations of the intermediate knowledge graph and the reasoning paths to give the user insight into how the system functions.

9.1.2 Visual Outputs for Guidance

We note that this entire work focuses on procedures being conveyed through text. Recently, there have been large advances in text-to-image generation models [32]. These models are now capable of providing incredibly realistic images if a given text prompt is good enough. These models could be leveraged for multi-modal procedural step generation. We give an example of this now utilizing the Dall-E model[143][1]

For the prompt:

> *Write the date in the line in the upper-right hand of a paper check. Use a wikiHow art style.*

In Figure 9-1 we see the result.

We note a couple of things, the Dall-E is a very general model for text-to-image generation. Additionally, there may not be training data in it for generating procedure-oriented clarification/representation images. While this particular image may not be useful, we note that for GUI descriptions, this may be more effective. Future work should explore combining visual descriptions of objects in steps to help with guidance. This was feedback that was given throughout the studies that we performed.

9.1.3 Multi-modal Inputs for Guidance

In the same way that we could hypothetically incorporate visual guidance outputs for our system, we could incorporate visual inputs to the model. This could help to guide people in the case that there are visually tasks that may require some kind of confirmation that a step that requires visual confirmation has been performed correctly. In other words, a user could take a picture of a result, and have the system confirm that it was the correct output. With GPT-4, which is capable of ingesting pictures, this could be tested out in future work.

---

[1] https://labs.openai.com

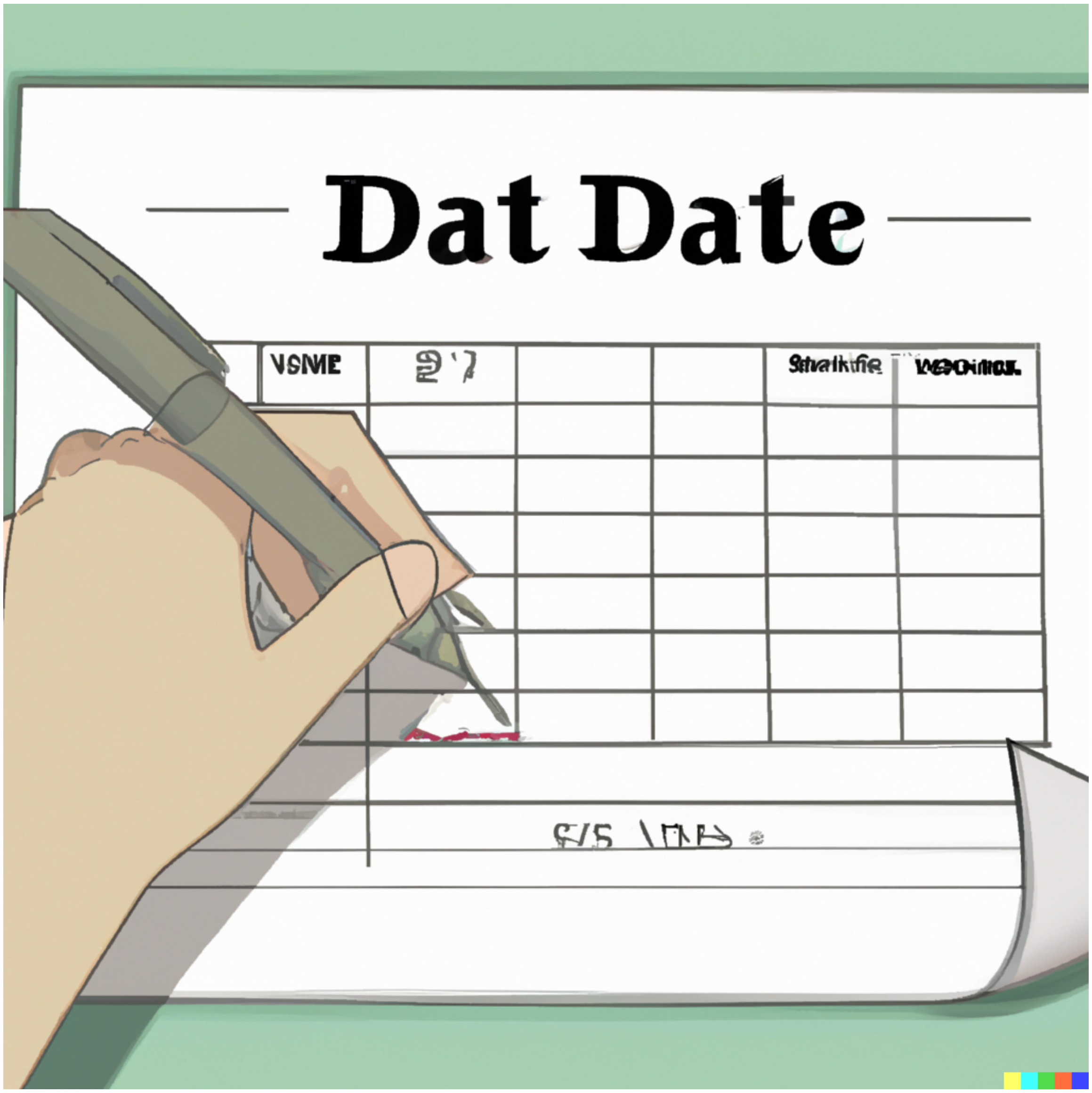


Figure 9-1: Dall-E's generation of prompt: *Write the date in the line in the upper-right hand of a paper check. Use a wikiHow art style.* We note that while it is a rough image, there is some semblance of a check and of writing something.

9.2 Final Remarks

We set out to build a conversational agent for helping people perform procedures that they may not know how to do, and having the ability to recover from errors if they ran into them while performing the procedures. We wanted to build such a system to help people understand and utilize technology. What began as a dream many years ago has now concretized into an agent that works and can be deployed. We wanted to make this agent rely solely on knowledge such that if needed it could explain its reasoning process clearly, and could combine its knowledge with new knowledge whenever it may be lacking to help with error handling. We now summarize everything that we did to make this dream come true.

In this fourth chapter, we describe the systems we developed to perform contextual, common sense inference for procedures. These systems utilize large language models, specifically transformer models, to ingest a story or procedure. We then generate a knowledge graph for each sentence or step of the procedure, by using a model that incorporates multiple knowledge sources and enables us to score generated assertions. Another unique contribution of our system is that it can be controlled, which allows us to apply it to generate facts about specific items. We also provide a proof-of-concept system that can extract procedural steps from web searches for a specific procedure. To train our step generation model and apply our reasoner, we contribute a wikiHow dataset with detailed steps, requirements and other data. This particular aspect of our work addresses the problem of providing contextual, common sense inference for procedures, and our contributions enable us to leverage this for procedural guidance later on.

In the fifth chapter, we demonstrate the possibility of generating steps for procedural guidance by formulating the problem as a graph-to-text task, where the graph is generated from a contextual commonsense inference model and the text represents the procedural step. We observe that the quality and amount of information significantly impact the generation system's performance. Based on this observation, we recommend a two-stage system with a planning stage and a generation stage, where the planner devises a useful plan for the generation process. We explore unsupervised, semi-supervised, and reinforcement learning systems for the planner but find that they underperform compared to a simple baseline of ranking knowledge based on the plausibility of the context. We also investigate using a large language model as a proxy for a reasoner, but find limitations in its contextual processing capabilities. Additionally, we devise a simple recursive algorithm to handle exceptions and issues within procedures, and explore the benefits of combining knowledge from different procedures to contextualize error handling.

In this sixth chapter, we introduce our conversational agent architecture consisting of a web-server backend and GUI frontend, which we utilize for user testing. Our primary agent employs an intent-based system that incorporates conversational patterns to convey procedural instructions. One contribution we make is to extend the telling conversational pattern to handle inquiries for error recovery, along with recursive extended tellings for this. Additionally, we present an agent powered by a large language model (LLM) that provides guidance through prompts, and a hybrid Knowledge Driven LLM that addresses usability issues discovered with the intent-based systems.

In the seventh chapter, we conduct two separate studies to evaluate the effectiveness of conversational agents in guiding users through procedures, including error recovery situations, and a third and final study to test a hybrid system approach. In the first study, we find that all three agents can successfully guide most users through procedures, although the intent-based agents require improvements in usability, particularly in intent-matching and slot-filling capabilities. We also observe that LLMs are highly capable of providing guidance in most procedures, but their information may be outdated.

In the second study, we assess the agents' ability to guide users in error recovery situations. We determine that all three agents can effectively guide individuals in such scenarios, but the intent-based agent struggles with understanding requests and requires improvement. Our findings highlight the importance of maintaining context during error interactions and emphasize the value of clear, concise step-by-step instructions in guiding users through procedures and error resolution. The LLM-based agent demonstrates greater flexibility in error handling and excels at providing guidance during exception processes.

In the third and final study, we leverage the LLM agent's flexibility in intent recognition, slot-filling, and chit-chat capabilities, combining it with our knowledge-driven step generation system and contextual commonsense inference facts. This hybrid system effectively identifies requests for procedural guidance, retrieves top-ranked facts and generated steps, and utilizes them as prompts to offer guidance to users. We observe that this hybrid system outperforms a baseline LLM in terms of providing more accurate and reliable knowledge. Additionally, we discover that incorporating descriptions of conversational patterns further improves the system's usability.

Based on our comprehensive research, we demonstrate that our contextual commonsense knowledge-based pipeline can effectively assist end-users through procedures, both in the form of an intent-based system and a more capable hybrid knowledge-LLM approach for procedure guidance and recovery.

In the eighth chapter, we discuss the limitations encountered during our work. Specifically, we address the limitations of large language models (LLMs) in providing procedural guidance for specific domains. While LLMs excel at offering general procedural guidance, their effectiveness diminishes in domains with limited documentation or insufficient training data. We also note their lack of adherence to conversational patterns, resulting in the generation of unhelpful responses for advancing a conversation. To overcome these issues, we propose informing the language model about the procedure at hand and extended telling to enhance its performance. Additionally, we explore leveraging our contextual commonsense inference system to provide the necessary information to LLMs for procedural tasks. Furthermore, we discuss the challenges in creating a flexible intent-based system, such as the one we utilized for general procedure handling, and propose leveraging LLMs as the intent recognition and interaction system to improve its capabilities. We also examine the limitations of our reasoner system and suggest that a more comprehensive knowledge retrieval system could enhance the ability to generate procedures on-the-fly.

In conclusion, we demonstrate that our contextual commonsense knowledge-based pipeline, integrated with conversational agents, is effective in providing procedural guidance and error recovery. We acknowledge the limitations encountered during our research and propose potential solutions to improve the performance of large language models and intent-based systems. Overall, our work contributes to the advancement of procedural guidance systems and highlights the importance of incorporating contextual facts and conversational patterns for more accurate and reliable assistance to users.

In summary, we present our work on developing a contextual commonsense inference system for procedures, which utilizes large language models and a knowledge graph to generate accurate and effective procedural guidance. We present an approach for generating procedural steps. We also developed a conversational agent architecture that incorporates conversational patterns to convey procedural instructions, and evaluated the effectiveness of our system through user testing. While we acknowledge some limitations in our work, we show the potential of our system in providing procedural guidance and error recovery for end-users. We also provide first insights into the use of more contemporary very large language models for this task of procedural guidance.

Overall, we find that we can do this task, and are hopeful that for the future, these agents will be deployed on a wide scale and utilized by everyone ranging from amateurs to experts for procedural guidance.

We make one final note. When we began research on this area around 5 years ago, such a system as the one that we proposed did not exist, nor did we have the slightest idea of how fast natural language processing would move with the advent of transformers. Although now it is possible to perform good zero-shot procedural guidance with very large language models, we have found that it is not perfect and has room to improve. We do not see this as obviating the need for work like ours, on contextual commonsense inference, nor on guidance, but as a complementary thing, a stepping stone in the right direction for providing guidance for the general audience.

9.2.1 Summary of Contributions

1. **Controllable Contextual Commonsense Inference System**

   A novel transformer-based, sequence-to system that is capable of taking a story, a sentence from that story, and a hint, and controllably produces a contextual commonsense assertion (fact) that is relevant to the story and sentence. The system is novel in that it utilizes the hint to guide the generation process, it uses a co-trained classifier to rate the plausibility of the assertions, and a single contextual commonsense inference model learns commonsense information from multiple aligned sources.

2. **Graph-to-Step System & Assertion Ranking Strategy**

   A novel transformer-based sequence-to-sequence system that takes a procedure name, a number of a step, prior steps, and a set of assertions and produces the indicated step for a procedure. This is the first system which applies the Data-to-text task for procedural step generation. This is combined with grounding the contextual inference process and sorting assertions by plausibility to select the essential knowledge for a procedural step.

3. **Conversational Procedure Error Handling Strategy**

   Novel utilization of an expanded Inquiry conversational pattern to handle procedural errors. Inquiries for error handling procedures are treated as Extended Tellings to provide error handling guidance. In addition to this, we combine contextual knowledge graphs to contextualize the generated steps for this error handling extended telling.

4. **Conversational agents for procedural guidance & error handling**

   Design and implementation of conversational agents that provide guidance with our novel contextual commonsense system. We show that the use of this conversational agent with the patterns along with our dynamic step generation system can be utilized to guide people through procedures and through error recovery in procedures.

5. **Hybrid Agent that uses Large-Language Models in combination with Graph-to-Step and Contextual Commonsense Inference**

   Implementation and evaluation of a novel conversational agent that leverages the flexibility of large language models and the explainability and knowledge-based approach of the contextual commonsense and graph-to-step system. We show that this hybrid system produces more accurate and reliable information that a plain large-language model. We also provide a first evaluation of large-language models applied to general procedure guidance

6. **Guidelines for Future Agent Development**

   From our evaluations and feedback we provide a list of guidelines that should be taken into consideration when developing conversational agents for procedures.

Appendix A

Study Questionnaires

A.1 Study 1

Here we show the questionnaire that we utilized for Study 1 (evaluating whether users could receive effective guidance from a system or not).

Procedural Guidance using Conversational Agents GPT

Start of Block: Intro Block

JS

Q1 **Thank you for participating in this study. An overview of how the study will work is the following:** First, you'll need to complete a pre-study questionnaire. Then, you'll be given a link to a conversational agent and asked to follow a specific procedure. Once you've interacted with the agent and attempted the procedure, you'll be asked to provide feedback on your experience. You'll repeat this process with another agent and provide feedback again. Finally, you'll complete a final questionnaire about your overall experience. Please be sure to read all instructions carefully to avoid having your task rejected and not receiving payment.
Disclaimer: *This study is part of a MIT scientific research project. Your decision to complete this study is voluntary. By completing the study, participants will share the audio from their conversation with the agent, transcripts of the conversation, and the time it takes to complete the process. If a participant is uncomfortable with this we ask that you do not accept the task. In this study some possible uncomfortable things are: although uncommon, strange responses from the voice assistant ranging from incoherent to possibly offensive voice responses. After completing the study, researchers will asses the results and reward participants within a week. Audio information will not be kept. There is no way for us to identify you. The only information we will have, in addition to your responses, is the time at which you completed the survey. The results of the research may be presented at scientific meetings or published in scientific journals. Clicking on the 'START' button on this page and the 'SUBMIT' button at the end of this study indicates that you are at least 18 years of age and agree to complete this study voluntarily.   Survey ID: ${e://Field/RandomId}(In case of questions)*

Q49 Write your Prolific ID (this will be used to link this survey with the Prolific one for validation and compensation)

____

End of Block: Intro Block

Start of Block: Pre-Study Block

Q1 How familiar are you with conversational agents (e.g. voice assistants like Siri, Alexa, Google Assistant)?

- ○ Not familiar at all (1)
- ○ Slightly familiar (2)
- ○ Moderately familiar (3)
- ○ Very familiar (4)
- ○ Extremely familiar (5)

Q2 What is your perception on the usefulness of conversational agents (e.g., Siri, Alexa, Google Assistant)?

- ○ Not at all useful (1)
- ○ Slightly useful (2)
- ○ Moderately useful (3)
- ○ Very useful (4)
- ○ Extremely useful (5)

Q3 In a typical day, how many times do you interact with a conversational agents (e.g., Siri, Alexa, Google Assistant)?

○ 0 times (1)

○ 1-2 times (2)

○ 3-5 times (3)

○ 6-10 times (4)

○ 11+ times (5)

JS

Q4 How many times have you performed these procedures in the past month

Never (0 times in the past month) | Rarely (1-3 times in the past month) | Occasionally (4-6 times in the past month) | Often (7-10 times a month) | Very Often (almost every day in the past month)

	0	1	2	3	4	5
Mirror a smartphone to a display wirelessly ()						
Share your location through text messages on a smartphone ()						
Compose an email in your smartphone with an attachment/embedded image ()						
Organize smartphone applications into folders ()						
Compose a Multi-media note in your smartphone ()						
Create a calendar entry in your smartphone with a video conference link ()						
Set up an automation on your smartphone ()						
Set up an email signature on your smartphone ()						

End of Block: Pre-Study Block

Start of Block: Agent1 Launch

Q6 Based on your previous answers you have been selected to perform the following procedure on your smartphone:

${e://Field/SelectedProcedure}

Make sure to adapt the procedure to devices that you have (for example, How do I edit a picture *on my iPhone/Android*? instead of How do I edit a picture on my smartphone) when asking for guidance.

To accomplish this, follow the link below. It will open a page to a conversational assistant. Please turn On/Up the Volume in your browser and utilize the microphone button to dictate your conversation. If needed you may type it also.

Remember, you have to ask this assistant for help/how to perform:
***${e://Field/SelectedProcedure}*.**
for a device you own.

Follow the agent's instructions on how to perform the procedure and attempt to perform the procedure.

Once you are done with this click next.

Note that you must log in utilizing the following credentials:

Username: ${e://Field/RandomId}
Password: ${e://Field/RandomId}

Also note that **the interaction with the agent will be recorded (audio and text transcripts). Failure to ask for instructions with the agent and follow through until the last step will result in invalidation of the task and no payment.**

Finally, note that the agent may give an error to wait a couple of minutes. Please wait and try again, this time has been taken into account when calculating the time for the study. If the error keeps happening, try to paraphrase your question. If the error still keeps happening, continue and message us.

${e://Field/AgentOpt1}" rel="noopener" target="_blank">Click Here to Launch the Agent Page

Optional: Upload an image with evidence that you did the procedure (This will expedite our review of the task)

End of Block: Agent1 Launch

Start of Block: Feedback 1

Q9 I felt the agent was conversational

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q10 I felt the conversation was natural

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q11 I had to pay special attention regarding my phrasing when communicating with the agent

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q12 I find that the agent understands what I want and helps me achieve my goal

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q13 The agent gives me the appropriate amount of information

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q14 The agent gave relevant information during the whole conversation

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q15 I thought the agent was easy to use

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q16 I would imagine that most people would learn to use this agent very quickly

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q17 I found the agent unnecessarily complex

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q18 I think that I would like to use this agent frequently

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Page Break

Q19 Did you succeed in your task (what the agent guided you on)?

○ Yes (1)

○ No (2)

Q20 Did you accomplish your goal (what you were told to do)?

○ Yes (1)

○ No (2)

Page Break

Q21 Did the agent misunderstand you? If so, how? Otherwise write N/A.

____

Q22 Did the interaction with the agent frustrate you? If so, how? Otherwise write N/A.

____

Q23 Did you encounter any problems when interacting with the agent? Otherwise write N/A.

____

Page Break

Q24 Did you encounter any problems when performing your task? If so, what? Otherwise write N/A.

____

Q25 What did you feel worked very well when completing the task?

____

Q53 What did you feel did not work well when completing the task?

____

Q26 What did you feel was useful or effective when interacting with the agent?

____

Q54 What did you feel was not useful or effective when interacting with the agent?

____

Q27 Anything else? How could this interaction be improved?

____

End of Block: Feedback 1

Start of Block: Agent2 Launch

Q52 Based on your previous answers you have been selected to perform the following procedure on your smartphone:

${e://Field/SelectedProcedure}

note that you may have to do the same procedure as before
Make sure to adapt the procedure to devices that you have (for example, How do I edit a picture *on my iPhone/Android*? instead of How do I edit a picture on my smartphone) when asking for guidance.

To accomplish this, follow the link below. It will open a page to a conversational assistant. Please turn On/Up the Volume in your browser and utilize the microphone button to dictate your conversation. If needed you may type it also.

Remember, you have to ask this assistant for help/how to perform:
${e://Field/SelectedProcedure}.
for a device you own.

Follow the agent's instructions on how to perform the procedure and attempt to perform the procedure.

Once you are done with this click next.

Note that you must log in utilizing the following credentials:

Username: ${e://Field/RandomId}
Password: ${e://Field/RandomId}

Also note that **the interaction with the agent will be recorded (audio and text transcripts). Failure to ask for instructions with the agent and follow through until the last step will result in invalidation of the task and no payment.**

Finally, note that the agent may give an error to wait a couple of minutes. Please wait and try again, this time has been taken into account when calculating the time for the study. If the error keeps happening, try to paraphrase your question. If the error still keeps happening, continue and message us.

${e://Field/AgentOpt2}" rel="noopener" target="_blank">Click Here to Launch the Agent Page

Q52 Optional: Upload an image with evidence that you did the procedure (This will expedite our review of the task)

End of Block: Agent2 Launch

Start of Block: Feedback 2

Q33 I felt the agent was conversational

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q34 I felt the conversation was natural

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q35 I had to pay special attention regarding my phrasing when communicating with the agent

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q36 I find that the agent understands what I want and helps me achieve my goal

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q37 The agent gives me the appropriate amount of information

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q38 The agent gave relevant information during the whole conversation

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q39 I thought the agent was easy to use

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q40 I would imagine that most people would learn to use this agent very quickly

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q41 I found the agent unnecessarily complex

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q42 I think that I would like to use this agent frequently

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Page Break

Q43 Did you succeed in your task (what the agent guided you on)?

- ○ Yes (1)
- ○ No (2)

Q44 Did you accomplish your goal (what you were told to do)?

- ○ Yes (1)
- ○ No (2)

Page Break

Q45 Did the agent misunderstand you? If so, how? Otherwise write N/A.

____

Q46 Did the interaction with the agent frustrate you? If so, how? Otherwise write N/A.

____

Q47 Did you encounter any problems when interacting with the agent? Otherwise write N/A.

____

Q48 Did you encounter any problems when performing your task? If so, what? Otherwise write N/A.

____

Q49 What did you feel worked very well when completing the tasks?

____

Q55 What did you feel did not work well when completing the tasks?

____

Q50 What did you feel was useful or effective when interacting with the agent?

____

Q56 What did you feel was not useful or effective when interacting with the agent?

____

Q51 Anything else? How could this interaction be improved?

____

End of Block: Feedback 2

Start of Block: Final Remarks

Q57 What is your perception on the usefulness of conversational agents (e.g., Siri, Alexa, Google Assistant)?

○ Not at all useful (1)

○ Slightly useful (2)

○ Moderately useful (3)

○ Very useful (4)

○ Extremely useful (5)

Q58 Do you think the tasks given were appropriate for this kind of agent? Why or why not?

____

Q48 Anything else? Is there any other feedback that you would like to give? This can be on the survey, the interactions, the interface, etc.

____

End of Block: Final Remarks

A.2 Study 2

Here we show the questionnaire that we utilized for Study 2 (evaluating whether users could recover from an error in guidance or not).

Procedural Guidance with Errors using Conversational Agents KDPG

Start of Block: Intro Block

JS

Q1 **Thank you for participating in this study. An overview of how the study will work is the following:** First, you'll need to complete a pre-study questionnaire. Then, you'll be given a link to a conversational agent and asked to follow a specific procedure. At a certain point in the procedure, you will have to utilize the agent to guide you to fix a mistake. Once you've interacted with the agent and attempted the procedure, you'll be asked to provide feedback on your experience. You'll repeat this process with another agent and provide feedback again. Finally, you'll complete a final questionnaire about your overall experience. Please be sure to read all instructions carefully to avoid having your task rejected and not receiving payment.
 Disclaimer: *This study is part of a MIT scientific research project. Your decision to complete this study is voluntary. By completing the study, participants will share the audio from their conversation with the agent, transcripts of the conversation, and the time it takes to complete the process. If a participant is uncomfortable with this we ask that you do not accept the task. In this study some possible uncomfortable things are: although uncommon, strange responses from the voice assistant ranging from incoherent to possibly offensive voice responses. After completing the study, researchers will asses the results and reward participants within a week. Audio information will not be kept. There is no way for us to identify you. The only information we will have, in addition to your responses, is the time at which you completed the survey. The results of the research may be presented at scientific meetings or published in scientific journals. Clicking on the 'START' button on this page and the 'SUBMIT' button at the end of this study indicates that you are at least 18 years of age and agree to complete this study voluntarily.   Survey ID: ${e://Field/RandomId}(In case of questions)*

Q49 Write your Prolific ID (this will be used to link this survey with the Prolific one for validation and compensation)

____

End of Block: Intro Block

Start of Block: Pre-Study Block

Q1 How familiar are you with conversational agents (e.g. voice assistants like Siri, Alexa, Google Assistant)?

- ○ Not familiar at all (1)
- ○ Slightly familiar (2)
- ○ Moderately familiar (3)
- ○ Very familiar (4)
- ○ Extremely familiar (5)

Q2 What is your perception on the usefulness of conversational agents (e.g., Siri, Alexa, Google Assistant)?

- ○ Not at all useful (1)
- ○ Slightly useful (2)
- ○ Moderately useful (3)
- ○ Very useful (4)
- ○ Extremely useful (5)

Q3 In a typical day, how many times do you interact with a conversational agents (e.g., Siri, Alexa, Google Assistant)?

○ 0 times (1)

○ 1-2 times (2)

○ 3-5 times (3)

○ 6-10 times (4)

○ 11+ times (5)

JS

Q4 How many times have you performed these procedures in the past month

	Never (0 times in the past month)	Rarely (1-3 times in the past month)	Occasionally (4-6 times in the past month)	Often (7-10 times a month)	Very Often (almost every day in the past month)	
	0	1	2	3	4	5
Share your location through text messages on a smartphone ()						
Compose an email in your smartphone with an attachment/embedded image ()						
Organize smartphone applications into folders ()						
Create a calendar entry in your smartphone with a video conference link ()						

End of Block: Pre-Study Block

Start of Block: Agent1 Launch

Q6 Based on your previous answers you have been selected to perform the following procedure on your smartphone:

${e://Field/SelectedProcedure}

Make sure to adapt the procedure to devices that you have (for example, How do I edit a picture *on my iPhone/Android*? instead of How do I edit a picture on my smartphone) when asking for guidance.

To accomplish this, follow the link below. It will open a page to a conversational assistant. Please turn On/Up the Volume in your browser and utilize the microphone button to dictate your conversation. If needed you may type it also.

Remember, you have to ask this assistant for help/how to perform:
***${e://Field/SelectedProcedure}*.**
for a device you own.

Follow the agent's instructions on how to perform the procedure and attempt to perform the procedure.

${e://Field/SelectedStop}

Note that you must log in utilizing the following credentials:

Username: ${e://Field/RandomId}
Password: ${e://Field/RandomId}

Also note that **the interaction with the agent will be recorded (audio and text transcripts). Failure to ask for instructions with the agent and follow through until the last step will result in invalidation of the task and no payment.**

Finally, note that the agent may give an error to wait a couple of minutes. Please wait and try again, this time has been taken into account when calculating the time for the study. If the error keeps happening, try to paraphrase your question. If the error still keeps happening, continue and message us.

${e://Field/AgentOpt1}" rel="noopener" target="_blank">Click Here to Launch the Agent Page

---

Optional: Upload an image with evidence that you did the procedure (This will expedite our review of the task)

End of Block: Agent1 Launch

Start of Block: Error 1

Q60 Now:

${e://Field/SelectedError}

Follow the agent's instructions to remedy the situation, and after the agent has guided you on attempting to fix this, try to complete the original procedure:

${e://Field/SelectedProcedure}

Once more note that the interaction with the agent will be recorded (audio and text transcripts). Failure to ask for instructions with the agent and follow through until the last step will result in invalidation of the task and no payment. Finally, note that the agent may give an error to wait a couple of minutes. Please wait and try again, this time has been taken into account when calculating the time for the study. If the error keeps happening, try to paraphrase your question. If the error still keeps happening, continue and message us.

Q61 Optional: Upload an image with evidence that you did the procedure (This will expedite our review of the task)

End of Block: Error 1

Start of Block: Feedback 1

Q9 I felt the agent was conversational

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q10 I felt the conversation was natural

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q11 I had to pay special attention regarding my phrasing when communicating with the agent

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q12 I find that the agent understands what I want and helps me achieve my goal

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q13 The agent gives me the appropriate amount of information

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q14 The agent gave relevant information during the whole conversation

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q15 I thought the agent was easy to use

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q16 I would imagine that most people would learn to use this agent very quickly

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q17 I found the agent unnecessarily complex

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q18 I think that I would like to use this agent frequently

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Page Break

Q19 Did you succeed in your task (what the agent guided you on)?

○ Yes (1)

○ No (2)

Q66 Did you succeed in the mistake fixing task (what the agent guided you on to fix the mistake)?

○ Yes (4)

○ No (5)

Q20 Did you accomplish your goal (what you were told originally to do)?

○ Yes (1)

○ No (2)

Page Break

Q21 Did the agent misunderstand you? If so, how? Otherwise write N/A.

____

Q22 Did the interaction with the agent frustrate you? If so, how? Otherwise write N/A.

____

Q23 Did you encounter any problems when interacting with the agent? Otherwise write N/A.

____

Page Break

Q24 Did you encounter any problems when performing your tasks? If so, what? Otherwise write N/A.

____

Q64 Were you able to recover from the problem that was given (${e://Field/SelectedError}) and finish the original task? Describe any issues or things that went well.

____

Q67 How would you ask a virtual agent for help if you made a mistake following instructions?

____

Page Break

Q25 What did you feel worked very well when completing the task?

____

Q53 What did you feel did not work well when completing the task?

____

Page Break

Q26 What did you feel was useful or effective when interacting with the agent?

____

Q54 What did you feel was not useful or effective when interacting with the agent?

____

Page Break

Q27 Anything else? How could this interaction be improved?

____

End of Block: Feedback 1

Start of Block: Agent2 Launch

Q52 Now you will perform the same task, with a different agent. Based on your previous answers you had been selected to perform the following procedure on your smartphone:

${e://Field/SelectedProcedure}

note that you may have to do the same procedure as before
Make sure to adapt the procedure to devices that you have (for example, How do I edit a picture *on my iPhone/Android*? instead of How do I edit a picture on my smartphone) when asking for guidance.

To accomplish this, follow the link below. It will open a page to a conversational assistant. Please turn On/Up the Volume in your browser and utilize the microphone button to dictate your conversation. If needed you may type it also.

Remember, you have to ask this assistant for help/how to perform:
${e://Field/SelectedProcedure}.
for a device you own.

Follow the agent's instructions on how to perform the procedure and attempt to perform the procedure.
${e://Field/SelectedStop}
Once you are done with this click next.

Note that you must log in utilizing the following credentials:

Username: ${e://Field/RandomId}
Password: ${e://Field/RandomId}

Also note that **the interaction with the agent will be recorded (audio and text transcripts). Failure to ask for instructions with the agent and follow through until the last step will result in invalidation of the task and no payment.**

Finally, note that the agent may give an error to wait a couple of minutes. Please wait and try again, this time has been taken into account when calculating the time for the study. If the error keeps happening, try to paraphrase your question. If the error still keeps happening, continue and message us.

${e://Field/AgentOpt2}" rel="noopener" target="_blank">Click Here to Launch the Agent Page

---

Q52 Optional: Upload an image with evidence that you did the procedure (This will expedite our review of the task)

End of Block: Agent2 Launch

Start of Block: Error 2

Q73 Now:

${e://Field/SelectedError}

Follow the agent's instructions to remedy the situation, and after the agent has guided you on attempting to fix this, try to complete the original procedure:

${e://Field/SelectedProcedure}

Once more note that the interaction with the agent will be recorded (audio and text transcripts). Failure to ask for instructions with the agent and follow through until the last step will result in invalidation of the task and no payment. Finally, note that the agent may give an error to wait a couple of minutes. Please wait and try again, this time has been taken into account when calculating the time for the study. If the error keeps happening, try to paraphrase your question. If the error still keeps happening, continue and message us.

Q74 Optional: Upload an image with evidence that you did the procedure (This will expedite our review of the task)

End of Block: Error 2

Start of Block: Feedback 2

Q75 I felt the agent was conversational

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q76 I felt the conversation was natural

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q77 I had to pay special attention regarding my phrasing when communicating with the agent

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q78 I find that the agent understands what I want and helps me achieve my goal

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q79 The agent gives me the appropriate amount of information

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q80 The agent gave relevant information during the whole conversation

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q81 I thought the agent was easy to use

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q82 I would imagine that most people would learn to use this agent very quickly

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q83 I found the agent unnecessarily complex

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q84 I think that I would like to use this agent frequently

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Page Break

Q85 Did you succeed in your task (what the agent guided you on)?

○ Yes (1)

○ No (2)

Q86 Did you succeed in the mistake fixing task (what the agent guided you on to fix the mistake)?

○ Yes (4)

○ No (5)

Q87 Did you accomplish your goal (what you were told originally to do)?

○ Yes (1)

○ No (2)

Page Break

Q88 Did the agent misunderstand you? If so, how? Otherwise write N/A.

____

Q89 Did the interaction with the agent frustrate you? If so, how? Otherwise write N/A.

____

Q90 Did you encounter any problems when interacting with the agent? Otherwise write N/A.

____

Page Break

Q91 Did you encounter any problems when performing your tasks? If so, what? Otherwise write N/A.

____

Q92 Were you able to recover from the problem that was given (${e://Field/SelectedError}) and finish the original task? Describe any issues or things that went well.

____

Q93 How would you ask a virtual agent for help if you made a mistake following instructions?

____

Page Break

Q94 What did you feel worked very well when completing the task?

____

Q95 What did you feel did not work well when completing the task?

____

Page Break

Q96 What did you feel was useful or effective when interacting with the agent?

____

Q97 What did you feel was not useful or effective when interacting with the agent?

____

Page Break

Q98 Anything else? How could this interaction be improved?

____

End of Block: Feedback 2

Start of Block: Final Remarks

Q57 What is your perception on the usefulness of conversational agents (e.g., Siri, Alexa, Google Assistant)?

○ Not at all useful (1)

○ Slightly useful (2)

○ Moderately useful (3)

○ Very useful (4)

○ Extremely useful (5)

Q58 Do you think the tasks given were appropriate for this kind of agent? Why or why not?

____

Q48 Anything else? Is there any other feedback that you would like to give? This can be on the survey, the interactions, the interface, etc.

____

End of Block: Final Remarks

A.3 Study 3

Here, we show the questionnaire that we utilized for Study 3 (evaluating whether a hybrid KDPSG and ChatGPT agent would provide more accurate/reliable procedures).

Procedural Guidance using Conversational Agents GPT-kGPT

Start of Block: Intro Block

JS

Q1 **Thank you for participating in this study. An overview of how the study will work is the following:** First, you'll need to complete a pre-study questionnaire. Then, you'll be given a link to a conversational agent and asked to follow a specific procedure on a iOS device. Once you've interacted with the agent and attempted the procedure, you'll be asked to provide feedback on your experience. You'll repeat this process with another agent and provide feedback again. Note that you will be given the same task twice. Finally, you'll complete a final questionnaire about your overall experience. Please be sure to read all instructions carefully to avoid having your task rejected and not receiving payment.

Disclaimer: *This study is part of a MIT scientific research project. Your decision to complete this study is voluntary. By completing the study, participants will share the audio from their conversation with the agent, transcripts of the conversation, and the time it takes to complete the process. If a participant is uncomfortable with this we ask that you do not accept the task. In this study some possible uncomfortable things are: although uncommon, strange responses from the voice assistant ranging from incoherent to possibly offensive voice responses. After completing the study, researchers will asses the results and reward participants within a week. Audio information will not be kept. There is no way for us to identify you. The only information we will have, in addition to your responses, is the time at which you completed the survey. The results of the research may be presented at scientific meetings or published in scientific journals. Clicking on the 'START' button on this page and the 'SUBMIT' button at the end of this study indicates that you are at least 18 years of age and agree to complete this study voluntarily.   Survey ID: ${e://Field/RandomId}(In case of questions)*

Q49 Write your Prolific ID (this will be used to link this survey with the Prolific one for validation and compensation)

____

End of Block: Intro Block

Start of Block: Pre-Study Block

Q1 How familiar are you with conversational agents (e.g. voice assistants like Siri, Alexa, Google Assistant)?

○ Not familiar at all (1)

○ Slightly familiar (2)

○ Moderately familiar (3)

○ Very familiar (4)

○ Extremely familiar (5)

Q2 What is your perception on the usefulness of conversational agents (e.g., Siri, Alexa, Google Assistant)?

○ Not at all useful (1)

○ Slightly useful (2)

○ Moderately useful (3)

○ Very useful (4)

○ Extremely useful (5)

Q3 In a typical day, how many times do you interact with a conversational agents (e.g., Siri, Alexa, Google Assistant)?

- ○ 0 times (1)
- ○ 1-2 times (2)
- ○ 3-5 times (3)
- ○ 6-10 times (4)
- ○ 11+ times (5)

JS

Q4 How many times have you performed these procedures in the past month

Never (0 times in the past month)	Rarely (1-3 times in the past month)	Occasionally (4-6 times in the past month)	Often (7-10 times a month)	Very Often (almost every day in the past month)

	0	1	2	3	4	5
Unsend a text message on iPhone ()						
Edit a sent message on iPhone ()						
Share location on messages in iPhone ()						
Change the clock style on the Lock Screen on iPhone ()						

End of Block: Pre-Study Block

Start of Block: Agent1 Launch

Q6 Based on your previous answers you have been selected to perform the following procedure on your smartphone:

${e://Field/SelectedProcedure}

To accomplish this, follow the link below. It will open a page to a conversational assistant. ***Please turn On/Up the Volume in your browser and try to utilize the microphone button to dictate your conversation. If the transcription is not very good, please type the message.***

Remember, you have to ask this assistant for help/how to perform:
***${e://Field/SelectedProcedure}*.**

Follow the agent's instructions on how to perform the procedure and attempt to perform the procedure.

Once you are done with this click next.

Note that you must log in utilizing the following credentials:

Username: ${e://Field/RandomId}
Password: ${e://Field/RandomId}

Also note that **the interaction with the agent will be recorded (audio and text transcripts). Failure to ask for instructions with the agent and follow through until the last step will result in invalidation of the task and no payment.**

Finally, note that the agent may give an error to wait a couple of minutes. Please wait and try again, this time has been taken into account when calculating the time for the study. If the error keeps happening, try to paraphrase your question. If the error still keeps happening, continue and message us.

${e://Field/AgentOpt1}" rel="noopener" target="_blank">Click Here to Launch the Agent Page

---

Optional: Upload an image with evidence that you did the procedure (This will expedite our review of the task)

End of Block: Agent1 Launch

Start of Block: Feedback 1

Q9 I felt the agent was conversational

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q10 I felt the conversation was natural

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q11 I had to pay special attention regarding my phrasing when communicating with the agent

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q12 I find that the agent understands what I want and helps me achieve my goal

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q13 The agent gives me the appropriate amount of information

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q14 The agent gave relevant information during the whole conversation

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q15 I thought the agent was easy to use

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q16 I would imagine that most people would learn to use this agent very quickly

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q17 I found the agent unnecessarily complex

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q18 I think that I would like to use this agent frequently

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Page Break

Q19 Did you succeed in your task (what the agent guided you on) with the agent?

○ Yes (1)

○ No (2)

Q20 Did you accomplish your goal (what you were told to do) with the agent?

○ Yes (1)

○ No (2)

Page Break

Q65 I think the guidance I was given is up-to-date

○ Yes (1)

○ No (2)

Q66 I feel like the chatbot’s responses were accurate.

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q68 I believe that the chatbot only states reliable information.

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q67 It appeared that the chatbot provided accurate and reliable information.

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Page Break

Q21 Did the agent misunderstand you? If so, how? Otherwise write N/A.

____

Q22 Did the interaction with the agent frustrate you? If so, how? Otherwise write N/A.

____

Q23 Did you encounter any problems when interacting with the agent? Otherwise write N/A.

____

Page Break

Q24 Did you encounter any problems when performing your task? If so, what? Otherwise write N/A.

____

Q25 What did you feel worked very well when completing the task?

____

Q53 What did you feel did not work well when completing the task?

____

Q26 What did you feel was useful or effective when interacting with the agent?

____

Q54 What did you feel was not useful or effective when interacting with the agent?

____

Q27 Anything else? How could this interaction be improved?

____

End of Block: Feedback 1

Start of Block: Agent2 Launch

Q52 Based on your previous answers you have been selected to perform the following procedure on your smartphone:

${e://Field/SelectedProcedure}

note that you have to do the same procedure as before

To accomplish this, follow the link below. It will open a page to a conversational assistant.***Please turn On/Up the Volume in your browser and try to utilize the microphone button to dictate your conversation. If the transcription is not very good, please type the message.***

Remember, you have to ask this assistant for help/how to perform: ***${e://Field/SelectedProcedure}.***

Follow the agent's instructions on how to perform the procedure and attempt to perform the procedure.

Once you are done with this click next.

Note that you must log in utilizing the following credentials:

Username: ${e://Field/RandomId}
Password: ${e://Field/RandomId}

Also note that **the interaction with the agent will be recorded (audio and text transcripts). Failure to ask for instructions with the agent and follow through until the last step will result in invalidation of the task and no payment.**

Finally, note that the agent may give an error to wait a couple of minutes. Please wait and try again, this time has been taken into account when calculating the time for the study. If the error keeps happening, try to paraphrase your question. If the error still keeps happening, continue and message us.

Note that you may have the transcript of the previous conversation, it is fine to ignore this, and the agent will not know this past conversation.

${e://Field/AgentOpt2}">Click Here to Launch the Agent Page

Q52 Optional: Upload an image with evidence that you did the procedure (This will expedite our review of the task)

End of Block: Agent2 Launch

Start of Block: Feedback 2

Q33 I felt the agent was conversational

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q34 I felt the conversation was natural

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q35 I had to pay special attention regarding my phrasing when communicating with the agent

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q36 I find that the agent understands what I want and helps me achieve my goal

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q37 The agent gives me the appropriate amount of information

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q38 The agent gave relevant information during the whole conversation

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q39 I thought the agent was easy to use

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q40 I would imagine that most people would learn to use this agent very quickly

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q41 I found the agent unnecessarily complex

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Q42 I think that I would like to use this agent frequently

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Page Break

Q43 Did you succeed in your task (what the agent guided you on) with the agent?

○ Yes (1)

○ No (2)

Q44 Did you accomplish your goal (what you were told to do) with the agent?

○ Yes (1)

○ No (2)

Page Break

Q61 I think the guidance I was given is up-to-date

○ Yes (1)

○ No (2)

Q62 I feel like the chatbot’s responses were accurate.

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q69 I believe that the chatbot only states reliable information.

○ Strongly disagree (4)

○ Somewhat disagree (5)

○ Neither agree nor disagree (6)

○ Somewhat agree (7)

○ Strongly agree (8)

Q64 It appeared that the chatbot provided accurate and reliable information.

- ○ Strongly disagree (4)
- ○ Somewhat disagree (5)
- ○ Neither agree nor disagree (6)
- ○ Somewhat agree (7)
- ○ Strongly agree (8)

Page Break

Q45 Did the agent misunderstand you? If so, how? Otherwise write N/A.

____

Q46 Did the interaction with the agent frustrate you? If so, how? Otherwise write N/A.

____

Q47 Did you encounter any problems when interacting with the agent? Otherwise write N/A.

____

Q48 Did you encounter any problems when performing your task? If so, what? Otherwise write N/A.

____

Q49 What did you feel worked very well when completing the tasks?

____

Q55 What did you feel did not work well when completing the tasks?

____

Q50 What did you feel was useful or effective when interacting with the agent?

____

Q56 What did you feel was not useful or effective when interacting with the agent?

____

Q51 Anything else? How could this interaction be improved?

____

End of Block: Feedback 2

Start of Block: Final Remarks

Q57 What is your perception on the usefulness of conversational agents (e.g., Siri, Alexa, Google Assistant)?

○ Not at all useful (1)

○ Slightly useful (2)

○ Moderately useful (3)

○ Very useful (4)

○ Extremely useful (5)

Q59 What kinds of procedures do you think these conversational agents would be helpful for?

____

Q60 What kinds of procedures do you think these conversational agents would NOT be helpful for?

____

Q58 Do you think the tasks given were appropriate for this kind of agent? Why or why not?

____

Q48 Anything else? Is there any other feedback that you would like to give? This can be on the survey, the interactions, the interface, etc.

____

End of Block: Final Remarks

Appendix B

Additional

B.1 Cosine Reasoner

Following our exploration of approximating the rouge score, we try to approximate and minimize the cosine distance between the embedding of a fact, a procedure title and the step number. We use a cosine embedding loss with confounders[1] to help maximize the cosine distance between the embedding of a fact from a different procedure, and a given procedure title and the step number. This serves a similar purpose as the rouge score, because high values of this score, will be more semantically similar to the step number of a certain procedure and therefore hopefully more relevant to the realization model. It is also a more relaxed problem in that it does not need to match exactly the step, it just needs to align with the step and procedure.

We tested this on a subset of 100 procedures, and from our results we found some interesting things. The first is that this model seems to converge on facts that are generally related to the procedure, but that may be irrelevant to the step at hand. The second is that when we look at the plans that it comes up with, we see that there are some useful facts for steps, but that it is still noisy. Although this is a promising reasoner, it is possible that some different kind of loss (e.g. a contrastive loss) should be used to try and remove the generally applicable facts, and keep the step-specific ones.

---

[1] https://pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html

B.2 Existing Procedural Interaction: Cooking

Before going on to design a conversational agent for procedural interactions, we looked at existing agents to see what the state of the art was for publicly available agents.

B.2.1 Heuristic Evaluation

We wanted to see how effective current conversational systems are at procedural tasks. To do this, we selected cooking as a procedural task and ran a heuristic evaluation of the interaction. We evaluated on an Amazon Alexa and on a Google Assistant the procedure of cooking rice. We took into consideration the criteria described in 6.3.

A summary of the results of the evaluation were the following:

- Navigation in the interaction is inconsistent, going back and forth and jumping around the steps in a procedure worked in different systems in different ways (jumping worked for a specific step, in other cases it moved forward or backward)
- No contextual inquiries, during the interaction: a user cannot ask for more information about the step or about anything given in the step
- No variation in levels of information and no rephrasing, the system always gives the same steps for the procedure, therefore if the person does not understand, then they cannot complete the procedure
- Repetitions are possible and a mere request for a repetition to the agent is handled appropriately
- Barging in works as expected stopping the agent from saying anything else
- No useful help, the help mechanism is not contextual to the procedural interaction. The help mechanisms, if present gave information about general interactions with the assistant
- Somewhat effective information breakdown, the systems make a good effort to parse and convey the procedure. The procedure that they parse has an appropriate length (usually a sentence or two per step)

- The initial start cooking intent is sometimes hard to nail down and may give a short answer, this means that a more guided process to start the interaction may be necessary

B.2.2 Small User Study for Usability of the Agents

We then ran this same interaction in a small user study (n=28) to validate that these concerns were present. In summary, we told users to cook rice using either an Amazon Alexa or a Google Assistant. In the study that was performed we found that the rice cooking application used a good, simple vocabulary and effective communication. We also found that the system lacks context of what is being discussed in the conversation and that it would benefit from having access to different levels of information. We found that the cooking interaction would benefit greatly from contextual question answering and needs to adhere to sequence closers (exiting when saying “enjoy" at the end of the recipe). We found that for the most part, the system had a good length of the utterances, although there are pieces that could be broken down further. We observed that most people tended to repeat and paraphrase themselves to get the assistant to understand them rather than ask the assistant to repeat or rephrase. We note that the cooking interaction system does not have the capability to paraphrase. Overall, it seems that for cooking, this basic system may be sufficient, and could be enhanced with a contextual question answering and paraphrasing system. This gave some hope that in other areas these agents might be useful too[2].

---

[2] For the complete study and its details, please check the following link

B.3 Additional Studies

B.4 Long-Term Open-domain Guidance

B.4.1 Overview

In the studies in Chapter 7, we evaluated whether a conversational agent can be used or not for guidance and if it helps people recover in procedures. For this study, we wanted to evaluate what domains people would use it in, and which domains may be better or not for this task. The overall results suggest that people tend to find simple, minimal complexity/ambiguity procedures, that procedural guidance agents should not only do procedures (i.e., they should have basic intent capabilities), and that even with robust intent detection it might not match the capabilities of a LLM powered agent.

B.4.2 Study Design

For this third study, we wanted to take a more open approach. We assigned each participant a particular agent. The options for the agent were the Google Assistant, the knowledge based procedural step generation system, or the ChatGPT-based system. We then told participants to use this agent for one week and ask it for guidance on whatever task they may need to do that requires instructions. After every interaction, participants were told to fill out a feedback form with the following questions:

- How would you rate your interaction (5 point Likert)
- What were you trying to do? (Open response)
- Did you succeed in your task (what the agent guided you on)? (Yes/No)
- Did you accomplish your goal (what you wanted to do)? (Yes/No)
- Did the agent misunderstand you? If so, how? If not write N/A (Open response)
- Did you encounter any problems when interacting with the agent? If so, what? If not write N/A (Open response)

- Did you encounter any problems when performing your task? If so, what? If not write N/A (Open response)
- If you encountered any problem, were you able to recover from the problem and finish the original task? If so, how? If not write N/A (Open response)

Our primary objective was to gather which domains procedural agents might be best suited or unsuited for. We also wanted to gather what was the performance of the Google Assistant, what we could call the most popular (or one of) state-of-the-art agent. We also wanted to see if there was any use of the exception handling mechanism in daily interactions.

We separated participants into 3 different groups and assigned them an agent to use either through an app or through a web-app. The Google Assistant users, if they had an iPhone, would download the Assistant app and in the case they had an Android phone, configure the on-device assistant to be the Google Assistant. For the KDPSG and ChatGPT-based agents, participants could either download an app for their device, or access the web-interface. In either case, the system worked in the same manner as the code was implemented in react-native and would have the same functionality in all the platforms.

In this study, we had 10 people participate, and the distribution was 3 people for the Google Assistant, 3 people for the ChatGPT agent, and 4 people for the KDPSG agent.

B.4.3 Study Results

Usability

We now describe the reported usability questions. We describe the “Instance Usability”, which is participants’ reported usability in each question, and the “Overall Usability”, which is the reported general usability in the post-study questionnaire.

Instance Usability We see in Figures B-3 and B-2 that the ChatGPT agent was able to satisfy most requests that people made to it, whereas the intent-based systems.

We see a similar, but distribution in how people rated the agent interactions in Figure B-1. From the feedback that we got, we saw that people were using the agents for more than just procedural tasks. We believe that this contributed to the KDPSG system receiving lower ratings for success and overall interaction rating. Additionally, the same intent/slot filling issues that came up in other studies happened here, which also possibly attributes to the lower score. We also see in the feedback that most people did not encounter an error in their interactions (108/163 recorded interactions or 66%), suggesting that they search for simple, probably unlikely to fail, interactions.

In the cases that there were errors, participants were able to recover from the problem and finish the original task by using common sense knowledge, asking again with different prompts, searching for information online, watching videos, or performing the task by themselves.

Overall Usability In the overall usability, we see a similar pattern as in the prior studies. However, we see that the Google Assistant follows the same, albeit slightly improved, perception as the intent-based KDPSG agent. We see that overall people that were assigned the ChatGPT agent, thought it was easier to use, more natural, conversational, and gave more relevant information during a conversation. We also see that the Google Assistant and the KDPSG agent have very similar distributions in all usability questions. This suggests, that if our KDPSG agent were made more robust, it might still not be capable of matching the ChatGPT agent in usability metrics. In the Figures presented below, the scale runs from 1 (Strongly Disagree) to 5 (Strongly Agree).

Areas of Usage

Summarizing the feedback from asking participants, “What areas/tasks do you think conversational agents would be good for?”, it suggests that these conversational agents are best suited for simple and precise tasks with straightforward and general solutions, such as, querying legal documents, cooking instructions, and tasks with pre-defined steps like installing software, hardware, or furniture. It can also be helpful for tasks

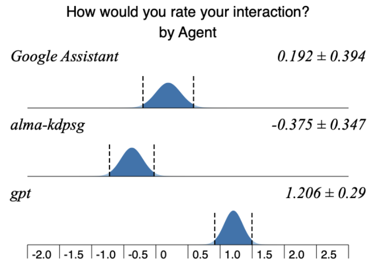


Figure B-1: Distribution of overall rating of interactions. We see the Google Assistant interaction rated as neutral, while the ChatGPT ones rated positively and the KDPSG ones rated slightly negatively.

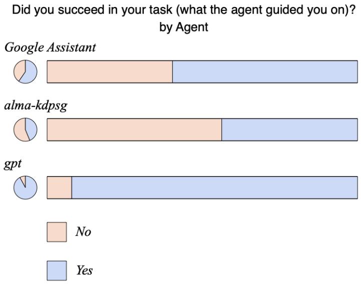


Figure B-2: Distribution of whether a participant succeeded in following steps given by the agent. We see that the Google Assistant had mixed results with half the people failing to follow what the assistant said, the KDPSG system trailing behind it, and that in most interactions people could follow what the ChatGPT agent said.

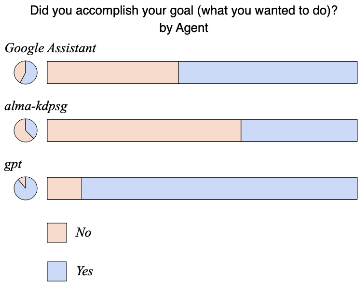


Figure B-3: Distribution of whether a participant succeeded in accomplishing their goals. We see a similar distribution to people's rating of the interaction. The Google Assistant satisfied half of the participant's goals, the KDPSG system satisfied slightly less than that, and the ChatGPT agent satisfied most people's goals.

that require common sense or involve learning a new thing.

However, the conversational agents may not be suitable for tasks that require sensorimotor skills, hands-on experience, or customized solutions. The feedback also indicates that the agents may not work for specific work-related issues such as permits or providing travel guidance. It may also not be suitable for tasks that are not straightforward or require visual guidance. Additionally, it may not be reliable for answering health-related questions unless it uses and can provide trusted sources. The agents may also not be helpful for tasks that require creative inputs or complicated

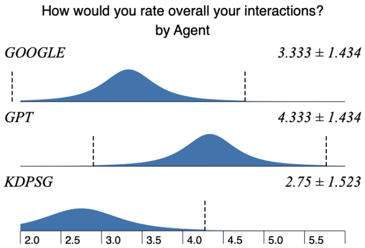


Figure B-4: Overall interaction rating. We see that the Google Assistant and the KDPSG assistant were rated close to neutral, and that the ChatGPT based agent was rated close to very good.

procedures.

We looked into the individual responses to find the following areas that people asked, in some form, for how-to guidance:

Cooking and Recipes: Several goals involve learning how to cook or prepare specific foods, such as scones, lamb kebabs, and alcoholic drinks. Additionally, some goals involve finding recipes, such as for enchilada sauce and banana chips.

Travel and Directions: Many goals involve finding information on how to get to a specific location or how to travel between two places. One example is finding the best time to visit Morocco.

Health and Wellness: Some goals involve learning how to perform specific health-related tasks, such as treating a sprained ankle, preparing for an MRI, and dealing with muscle soreness. Some people also wanted to find out how to improve their sleep and organize their day.

Technology and Devices: Several goals involve learning how to use or troubleshoot various devices, such as resetting a Samsung Smart TV, downloading an app on a TV, and activating a new phone.

Personal Finance: A few goals involve learning how to perform financial tasks, such as filing taxes. Additionally, some people looked at how to cancel a debit card.

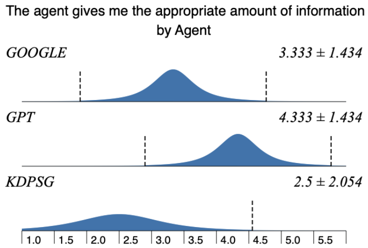


Figure B-5: Distribution of whether an agent gave an appropriate amount of information or not. We see that people were neutral on intent-based agents and agreed on the ChatGPT agent.

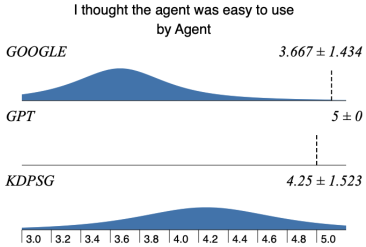


Figure B-7: Distribution of whether an agent was easy to use or not. Most people leaned towards agreeing that all agents were easy to use.

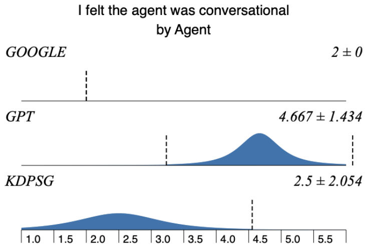


Figure B-6: Distribution of whether a participant found the agent conversational or not. Most participants agreed that the ChatGPT agent was conversational and disagreed that the intent based ones were.

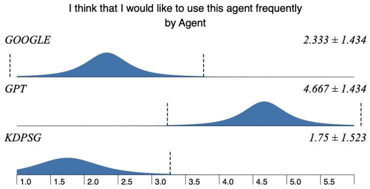


Figure B-8: Distribution of whether a participant would use the agent frequently or not. People agreed that they would use the ChatGPT agent frequently and disagreed on the intent-based ones.

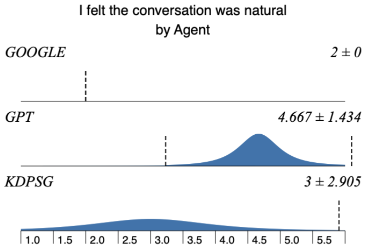


Figure B-9: Distribution of whether the conversation felt natural or not. We see that the intent-based agents were rated neutral to disagree and that the ChatGPT one was rated higher than agree.

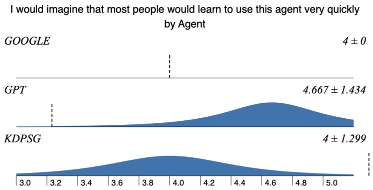


Figure B-11: Distribution of whether participants would learn to use the agent quickly or not. Most people leaned towards agreeing that all the agents could be learned quickly.

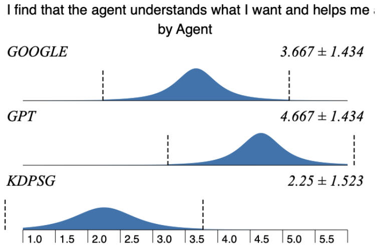


Figure B-13: Distribution of whether participants thought that the agent understood them and helped them achieve their goals. We see that the KDPSG agent was rated as the least understanding/useful for goal accomplishment, and the ChatGPT and Google Assistant being rated higher. This once more suggests that the intent mechanism needs to be made robust.

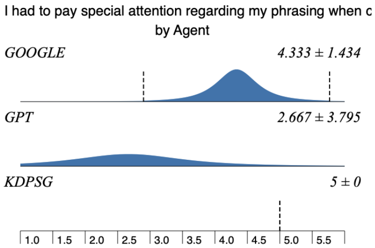


Figure B-10: Distribution of whether a participant had to use special phrasing or not. Most people found that they needed to use special phrasing with the intent-based agents and leaned towards disagreeing on the ChatGPT agent.

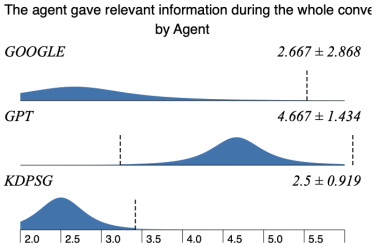


Figure B-12: Distribution of whether an agent gave relevant information or not. The ChatGPT agent was rated as giving the most relevant information, and the intent based agents people leaned towards disagreeing.

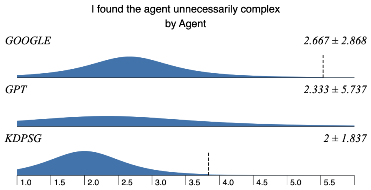


Figure B-14: Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people disagreed that the agents were complex.

General Education and Learning: Some goals involve learning how to do specific things, such as tie a tie, fold a fitted sheet, and draw a face.

We make an interesting note that while people were instructed in the onboarding to ask/report only behaviors for how-tos, people also reported some general assistant queries such as currency conversions, definitions, and general factoids, among others. What this suggests is that, the ability to perform procedural guidance should be added as one of these basic/default intents to agents or vice versa, that a procedural guidance agent should have the ability to handle other general open requests.

B.4.4 Overall Findings

We summarize the findings of this study:

- The results suggest that procedural guidance should be a complementary additional behavior in conversational agents, and not an exclusive behavior as people query for a wide variety of possibly non-procedural things.

- The results suggest that procedural guidance agents seem to be likely to have good effects on what people called "simple" procedures. Expanding on this, a simple procedure should be clear and concise, with minimal complexity or ambiguity, so that it can be easily executed by someone with basic knowledge or skills.

- The results suggest that people do not expect the agent to work in highly specialized domains such as medicine, or scientific fields. That more verification and sourcing mechanisms need to be added to make these agents work in these specialized domains.

- The results suggest that agents would be very effective in technology related fields such as device usage, setup, and troubleshooting. From these results and feedback from other studies, people expect that the agents that are on smartphones know the ins and outs of the device that they run in.

- The results suggest that in most interactions, participants did not encounter an error in the process that they were performing, reinforcing the idea that they only search for simple, unlikely to fail, procedures

B.5 Closed Domain (in-person) Guidance

B.5.1 Overview

For our last study, since we ran the first two studies on an online platform, we wanted to get a hands-on, in-depth study of how people interacted with the agents. We reused the same task from Study 2, to be able to see how people deal with doing a procedure and fixing an error in it. In contrast to study two, we have users alternate between using the knowledge driven approach, and the GPT based agent.

B.5.2 Study Design

We have the same design as in study two, people were given a form to fill out from which a certain task would be picked, and the person would have to do that task. Once more, in comparison to study 2, people were given the KDPSG agent, and the GPT agent rather than the gold instruction retrieval baseline. As participants went through the form, we ask them questions such as: would you be able to do this if it was purely conversational (i.e., if there was no screen), we also asked: “what happened?”, whenever they hit an error, and we asked them to explain whenever we noticed that they were surprised or frustrated by something. We also try to enforce as much as possible, the participants actually stopping while in an interaction, and receiving the news that they had made a mistake. In this study, we had 9 participants.

B.5.3 Study Results

In this study, we wanted to focus more on the qualitative feedback of the experience. We found that in regard to frustration and misunderstandings for KDPSG, the agent struggled to understand specific details and context, leading to frustrations for the participants. The participants had to be very specific in their queries for the agent to understand, and there were instances where the agent did not pick up on follow-up questions or forgot the history of the chat. We noted, during all the interactions, that users ran into intent recognition problems. Most users were able to overcome it and rephrase into something the agent understood. When asked during the failure, some participants mentioned that they noticed that they had to be very precise in what was said so that the agent could pick up the intent correctly and perform the slot-filling correctly. It is also worth noting that every time intent-based agent kept asking for a rephrase, people would sometimes laugh in frustration. We would also see that the phrasing when declaring that an error occurred, participants would sometimes phrase it as a description of the error that had occurred. For intent-recognition and slot filling, this would also be a complicated task, because firstly how does one determine if there was an error from a statement, and secondly how do we know that the user is asking for help with the error. Contextual cues would help in this, however the current intent-recognition does not use past history, exacerbating the issue.

Overall, we could see very clearly that the intent system needed to be considerably more robust than what was implemented. Although it may be possible to address this in these studies, considering the semi-limited amount of procedures that could have been done, in a real world scenario this may be more complicated. If we wanted truly open-domain guidance, it would need to be much more robust for the slot filling to be able to capture or discern what the person wants to accomplish.

With the ChatGPT agent, participants never complained about the agent not understanding them. It was able to provide an adequate response, even with typing errors in the prompt. Most of the ChatGPT agent interactions went smoothly. People were able to communicate with it in simple terms and would get the correct/expected response. In one particular case, the participant found that the agent provided outdated information. The participant kept probing the agent trying to fix the issue, and was ultimately able to figure out how to do it by themselves. The participant corrected the ChatGPT-based system, but even with the correction, the system kept reusing its old information. In this probing that the participant did, we could find that in the case that the agent did not know how to perform the procedure, it could make up procedures to seem as if it were able to fix the issue. We noted these behaviors, and based on this interaction, we were able to design the final study that we ran.

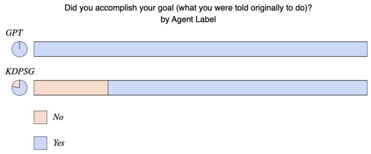


Figure B-15: Distribution of participants that were able to accomplish their goal. We see that most people were able to with both agents.

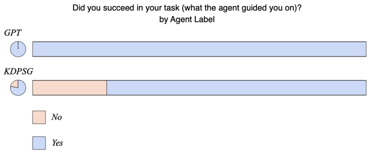


Figure B-16: Distribution of participants that were able to complete the task that the agent gave them guidance. Once more, we see that most people were able to accomplish what the agent suggested doing

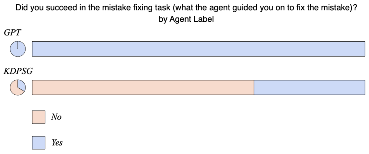


Figure B-17: Distribution of interactions that were able to recover from errors. We see that with the ChatGPT based agent most people could recover, however with the intent-based agent not many people could recover. In the interactions, and from the usability feedback, we see that this in large part is due to lack of understanding in the intent detection/slot filling

B.5.4 Overall Findings

We now list some of the findings that we observed.

- The results suggest that the intent-recognition and slot-filling issues were pervasive in the intent-based agent.
- The results show that people were still able to utilize the agent for guidance, even in the face of intent recognition problems.
- The results show that people struggled to recover from the errors if the intent-detection failed to understand their phrasing of the error.

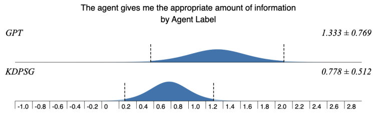


Figure B-18: Distribution of whether an agent gave an appropriate amount of information or not. We see that the participants find that both agents give an appropriate amount of information.

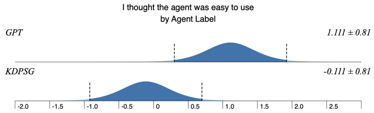


Figure B-20: Distribution of whether an agent was easy to use or not. Most people leaned towards agreeing that the ChatGPT agent was easy to use and were neutral, leaning to negative on the KDPSG..

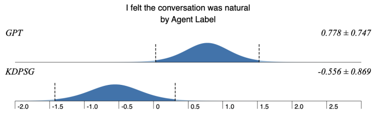


Figure B-22: Distribution of whether the conversation felt natural or not. We see that leaned towards disagree for the KDPSG and agree for the ChatGPT agent.

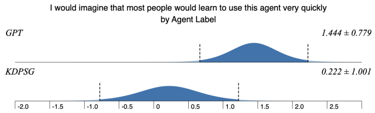


Figure B-24: Distribution of whether participants would learn to use the agent quickly or not. People agreed that the ChatGPT agent would be quick to use and were neutral leaning to positive on the KDPSG.

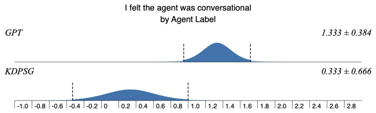


Figure B-19: Distribution of whether a participant found the agent conversational or not. Most participants agreed that the ChatGPT agent was conversational and were neutral on the KDPSG.

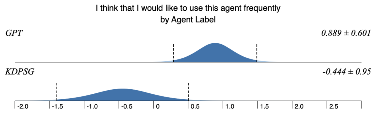


Figure B-21: Distribution of whether a participant would use the agent frequently or not. People agreed that they would use the ChatGPT agent frequently, and leaned towards disagreeing on the KDSPG.

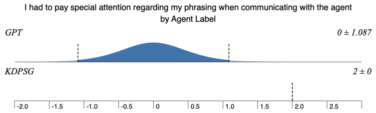


Figure B-23: Distribution of whether a participant had to use special phrasing or not. Most people found that they needed to use special phrasing with the intent-based agents and leaned towards disagreeing on the ChatGPT agent.

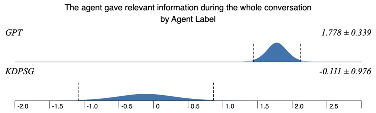


Figure B-25: Distribution of whether an agent gave relevant information or not. The ChatGPT agent was rated as giving the most relevant information, and neutral on the KDPSG.

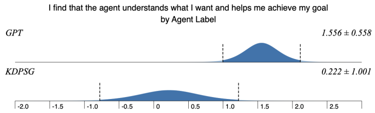


Figure B-26: Distribution of whether participants thought that the agent understood them and helped them achieve their goals. We see that the KDPSG agent was rated neutral leaning to positive, and the ChatGPT based agent was rated as, agree.

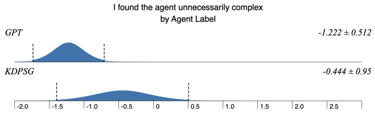


Figure B-27: Distribution of whether a participant thought that an agent was unnecessarily complex or not. Most people disagreed that the agents were complex.

Appendix C

Related Works

C.1 Related Work for Abstracting Procedures to Knowledge Graphs

C.1.1 Commonsense: Grounding, Reasoning, and Knowledge

A related line of work has been in grounding commonsense statements for inference. However, this line of work is more aligned with natural language inference rather than assertions. One contribution in this area is HellaSwag [187] which constructs a question-answering dataset whose plausible answers are intended to be confounders to language models. Our work differs from this line of work in that we intend to produce structured outputs.

Other work looks at reasoning with commonsense knowledge graphs. One work that utilizes the explicit graph structure to perform multi-hop reasoning is "Commonsense for Generative Multi-Hop Question Answering Tasks" [91]. The authors look to select grounded multi-hop relational commonsense information from ConceptNet via a pointwise mutual information and term-frequency based scoring function to fill in gaps of reasoning between context hops for a model they use. In contrast to this work, we are not looking at the task of question answering.

Some older work that looks at doing something similar to what we define as joint inference is blending [58]. This technique essentially consists of constructing and adding or blending together matrices of embeddings to find the commonalities between discrete knowledge sources and commonsense knowledge. This method, however, is hard to scale to large knowledge bases and is not easily applied to the task of contextual knowledge inference. An even older project that looks into a certain kind of joint inference is Cyc [77]. Cyc uses the idea of "micro-theories"; there would be a small set of commonsense assertions that you could reason with, then combine them with the more general Cyc KB. However, this is not really joint inference in our context, but rather trying to address the problem of local vs. global inference.

We examine also the work by [23] which utilizes formal logic and restructuring of assertions to be able to combine and perform joint inference over knowledge bases, however, the system as it is cannot be used for contextual commonsense inference because it requires explicit knowledge to be already present in a knowledge base to make inferences, whereas in our work through the underlying language models can produce inferences for unseen concepts. Additionally, we look at the work [120], which uses a combination of systems to extract high-quality non-triple formatted facts from the text. However, the system is based on grammatical structures in an input text, which means that implied facts may not be extracted from this if such a system were utilized for contextual commonsense inference.

Other works that have tried to consolidate commonsense knowledge are the following. [66] examines multiple sources of knowledge and unifies the relations in these under 13 dimensions of commonsense, however, it still remains a challenge to unify the nodes in the different sources, and such a broad unification may make it challenging to generate inferences for detailed relations (i.e., a specific relation type).

Lastly, we mention some works on open knowledge bases that could be leveraged in future work for utilization in joint inference: TupleKB [104], Quasimodo [154], Ascent [121], GenericsKB [12].

C.1.2 Adversarial Language Models

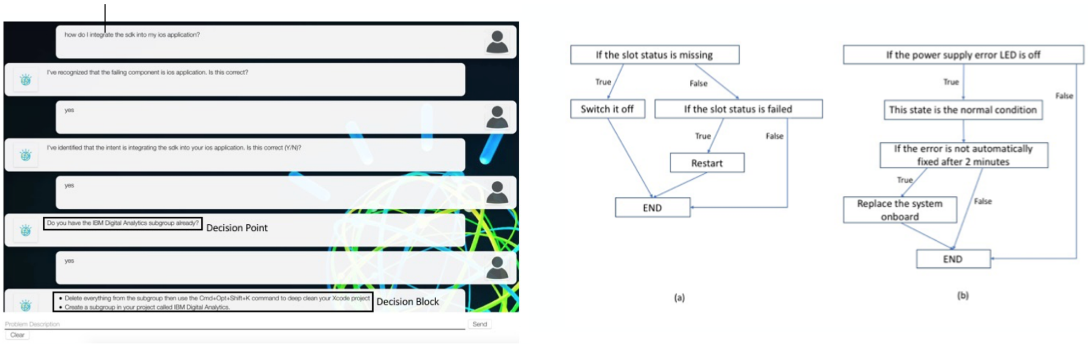
Here we look into work that utilizes adversarial or pair training with language models. One such work is [178] in which the authors utilize a GPT-3 [19] model as a teacher in order to distill commonsense knowledge into a student model that is considerably smaller. This task is different from ours in that they do not explore contextual/discourse-aware commonsense inference, instead, they look at extracting the knowledge already found in a model.

Other work more similar to ours, albeit older, is [136]. In this work, the authors take a similar approach to our adversarial configuration, however, they utilize the Wasserstein GAN objective [7] rather than the basic GAN formulation that we use. The authors additionally use the same approximation that we utilize in section 4.5.2. We note that they employ other strategies such as teacher helping, curriculum learning, and variable length that are worth looking into for future work. We also note that the authors tackle general language generation, rather than our task of contextual commonsense inference.

C.2 Related Work for Reasoning and Generation of Procedural Steps

C.2.1 Procedural Knowledge Extraction and Understanding

We start with DESIRE [56], a system that was built by IBM to mine procedures from their support website. As an attempt on improving the handling of "How-to" questions, IBM performed a study on a set of support tickets that they had. It turned out that 26% of the tickets that they analyzed were about people directly seeking a procedure [56]. To attempt to address this, they built an agent that helps people access procedures by chatting. The agent they built would extract the steps in the procedure, and build an instruction flow-graph-like structure that could be traversed to relay a procedure through a chatbot. The system that we present in this broader work (i.e. Winston), defers from this primarily in that we will be generating a full intermediate knowledge graph, giving extra information on principles/relations and properties of the entities that appear in the procedure, not only on the high-level procedure text. Additionally, the system we give guidance on would be able to support just-in-time exceptions in the procedure by splicing in additional knowledge and uses a reasoner to generate the conversational text from a walk through an intermediate graph rather than using the HTML structure.



how do I integrate the sdk into my ios application?

I've recognized that the failing component is ios application. Is this correct?

yes

I've identified that the intent is integrating the sdk into your ios application. Is this correct (Y/N)?

yes

Do you have the IBM Digital Analytics subgroup already?

yes

- Delete everything from the subgroup then use the Cmd+Opt+Shift+K command to deep clean your Xcode project
- Create a subgroup in your project called IBM Digital Analytics.

Decision Point
Decision Block
Problem Description
Send
Clear
If the slot status is missing
True
False
Switch it off
If the slot status is failed
True
False
Restart
END
(a)
If the power supply error LED is off
True
False
This state is the normal condition
If the error is not automatically fixed after 2 minutes
True
False
Replace the system onboard
END
(b)

Figure C-1: DESIRE Chatbot Interaction (left), DESIRE Intermediate Procedural Representation (right)[56]

The aforementioned work on the DESIRE chatbot builds a flow graph of a procedure. This can be seen as a high-level understanding of what happens across the steps. It, however, does not have a low-level understanding of the entities and their states within the individual steps. The DESIRE system uses a search system combined with a classifier to determine where the steps in a procedure are located. The classifier in their system utilizes features such as the number of sentences or steps with at least one imperative, the number of sentences or steps that start with an imperative, and the density of extracted imperatives in text, to determine whether an HTML formatted list is a procedure or not.

When they find the instructions, they take the ordering as-is to construct their instruction flow graph. They also highlight decision points, which are points where, depending on a condition, the procedure forks to a different path[56]. Formally, we give the notation that every sentence s_t and each entity e_l in a procedural document PD. A procedural document can be seen as a text that contains procedural instructions. The DESIRE system builds a tree-like structure with the ordered procedure sentences found in PD with $|PD| = 1$. We additionally use the notation of $PS = \{s_1, s_2, ..., s_t, ..., s_{T-1}, s_T\}$, $PS \subset PD_i$; $i = 1...k$ for the T sentences contained in a procedural document PD_i. The system that powers DESIRE, although it abstracts the procedure into a tree-like, flow graph, gives no insight as to how people could handle possible problems that occur during steps. The nodes in this graph are sentences from PS and the edges are directional towards the next sentence this can be seen in Figure C-2 (i.e. $s_1 \rightarrow s_2$ and so on unless there is a decision point, in which case it would go from a sentence to the possible paths).

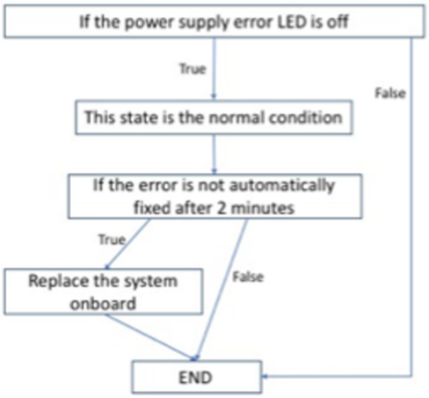

If the power supply error LED is off
True
False
This state is the normal condition
If the error is not automatically fixed after 2 minutes
True
False
Replace the system onboard
END

Figure C-2: Flow graph generated by DESIRE[56]

Explaining Action Dependencies (XPAD) [103] is a model which tries to understand dependencies between steps and how these change the states of entities. For every sentence s_t and each entity e_l in a procedural document PD, an encoder creates a contextual vector c_{tl}, capturing how the actions in s_t affect e_l. The vectors c_{tl} are then handed to a decoder that determines what state changes (create, move, destroy, none) has the entity undergone. The decoder then performs a beam search of possible decoding options and selects one that is globally consistent based on a scoring function. In their work, they define the scoring function as $\phi(\pi_t, \mathcal{G}_t)$, where π_t and $\mathcal{G}_t$ are the state change matrix for step s_t and the dependency graph between steps up to step s_t respectively. The decoder's output is a combination of the state matrix along with a dependency graph of the steps. The dependency graph can be plotted out to give more clarity on what the understanding model believes to be true the correct sequence of events for the explainability of the causes or requirements of steps. An example of this can be seen in figure C-3.

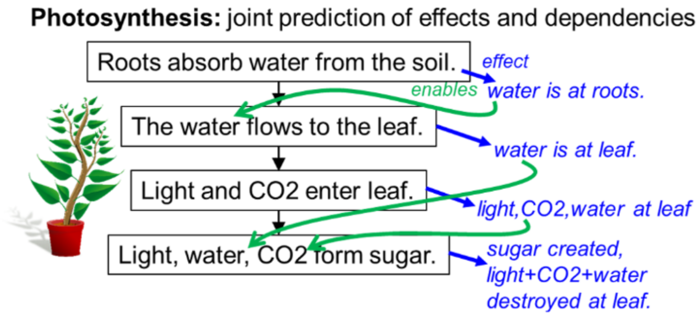

Photosynthesis: joint prediction of effects and dependencies
Roots absorb water from the soil.
effect
enables
water is at roots.
The water flows to the leaf.
water is at leaf.
Light and CO2 enter leaf.
light,CO2,water at leaf
Light, water, CO2 form sugar.
sugar created,
light+CO2+water
destroyed at leaf.

Figure C-3: Dependency visualization for XPAD system [103]

DynaPro [5] goes one step further and incorporates a contextual language model (BERT [40]) to further understand how entities change through steps. The authors develop an end-to-end neural procedural reading comprehension model that jointly identifies entity attributes and transitions, leveraging dynamic contextual encoding of the procedural text. DynaPro starts by getting the contextual representation of each entity at every time step, then it identifies entity attributes for current and previous time steps and builds an attribute-aware representation of the procedural context. Lastly, it uses the contextual entity representation and the attribute representation to state transitions at a time step.[5] If we utilize our notation, and try to keep in line with XPAD, DynaPro generates a different contextual timestep-entity vector c_{tl} by using the representation provided by BERT up until the timestep t.

Additionally, they use this BERT representation to find attributes that appear in the text and encode them along with the BERT representation. Keeping with our notation, we call what they refer to as the entity and attribute aware representation the modified contextual vector c'_{tl}, Additionally, rather than using a decoder with a beam search as in XPAD, they use an LSTM that takes into account this modified context vector, c'_{tl} to predict state transitions for the attributes. An interesting feature of this system is that it is able to pick up entity attributes directly from the procedural text and track them as the steps go on. If combined with an entity linker, this system could potentially pull in external information by replacing the BERT model with

KnowBERT[128] to incorporate encoded entities, although more research would be required on this idea.

XPAD and Dynapro work by only looking at extracting the procedures from one procedural document (i.e. $|PD| = 1$). They also do not incorporate any external world information that could make their tasks easier. Lastly, these works offer no insight as to how we could stitch in a sub-procedure in the case that something went wrong. These systems serve to give us an insight into the internal mechanisms of procedures (i.e. how things are created/moved/destroyed/located), however, they are disconnected from world knowledge such as the commonsense information found in knowledge graphs such as ATOMIC [155], ConceptNet [162], and more recently GLUCOSE [113]. This could hinder the systems' understanding of how the entities in the procedure interact with the world and in addition to this, may hinder the generalization of understanding of entities' states to previously unseen entities, however, more experimentation is needed to confirm this.

We highlight in this space the work by Li Zhang **??????**. Zhang et al. have developed their work concurrently, and has shown very similar things to some of the points that we make in this work, in particular that knowledge is essential to procedural understanding, and in particular, procedure knowledge needs to be extracted and structured well to be able to perform inference for next steps [189]. We could see our contextual commonsense inference as a superset of this information extraction. Zhang proposes many interesting ideas and gives an excellent summary of the current state of procedural understanding [188].

C.2.2 Commonsense and Procedures

Commonsense information refers to the millions of basic facts and understandings possessed by most people [92]. There is some evidence that the incorporation of commonsense information can be useful in procedural understanding in ProStruct [167]. The authors take a handful of commonsense facts (e.g. an entity cannot be created if it already exists) and use it to filter out impossible or improbable states from an encoder-decoder system. The authors mention that: "The commonsense constraints

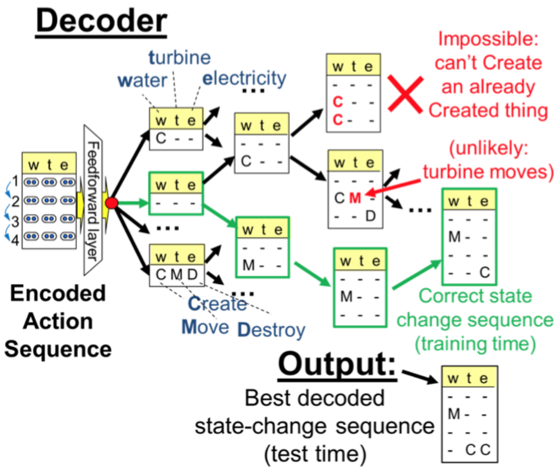

Decoder
turbine
water
electricity
Feedforward layer
Encoded Action Sequence
Create
Move
Destroy
Impossible: can't Create an already Created thing
(unlikely: turbine moves)
Correct state change sequence (training time)
Output:
Best decoded state-change sequence (test time)

Figure C-4: ProStruct decoder architecture that utilizes commonsense constraints to filter out impossible/improbable states [167]

we have used for ProPara[1] are general, covering the large variety of topics contain in ProPara, [however to use it for] other genres of procedural text, or [to] broaden the state change vocabulary, different commonsense constraints may be needed". These constraints could be injected from one of the commonsense knowledge graphs that we have mentioned (ConceptNet, ATOMIC, GLUCOSE). With our prior work on contextual commonsense inference, this information could be even generated on the fly. Keeping in line with our aforementioned notation, this work would modify XPAD in that the scoring function from XPAD: $\phi(\pi_t, \mathcal{G}_t)$, would be modified to include a set of commonsense constraints C_{cs}: $\phi'(\pi_t, \mathcal{G}_t, C_{cs})$ We discuss knowledge graphs and their use in procedures in the next section.

The work by Ribeiro et al. [147]. also shows that by leveraging commonsense information, one can generalize better the ability to infer how an entity behaves in a procedure for unseen entities. The work builds a collection of relevant "query cases" which are "link[s from] annotated answers to semantic interpretations (i.e. logical statements)". These query cases are essentially mappings of text examples to semantic assertions. When the system performs an evaluation, it retrieves these

---

[1] ProPara is a dataset designed to train and test comprehension of simple paragraphs describing processes for the task of predicting, tracking, and answering questions about how entities change during the process[102]

query cases to solve a question through analogy like reasoning (i.e. if the query cases give evidence of the answer, then it is likely that the question at hand has a similar answer).

Finally, we look into some of the results of a collaboration [126]. The author explored incorporating symbolic information, in the form of knowledge graph triples, into the task of Open-Domain prediction of state changes in procedures [168]. Some of the findings are that relationships that stem from highly useful or relevant entities in the procedure, help in this task of procedural understanding of state changes. Additionally, the author finds that incorporating the Goal of the procedure, helps the model infer state changes in the entities involved. Both of these findings suggest that procedural understanding requires information that may not be explicit in the procedure, such as extra knowledge, and at least some formulation of the final goal of the procedure.

C.2.3 Procedure Representations

We have described knowledge graphs to represent commonsense information, but they can also be used to represent and analyze procedures. We have already described DESIRE [56] and how it generates a flow-graph of steps in a procedure. Although not explicitly a knowledge graph, it could be interpreted as one, by giving the edges a label such as “hasSubEvent" or "leads to" similar to ConceptNet. We now describe other works that have used graphs to work with procedures.

The first is “Distilling Task Knowledge from How-To Communities” [26]. In [26] the authors build *task frames* which are a task phrase[2] enhanced with attributes (e.g. location, time, participating agent, participating object, category), hierarchical relations (e.g. parent task and sub-tasks), and temporal relations (e.g. previous task and next task). The authors use OpenIE [46] systems to extract the frames along with

---

[2] A phrase consisting of (v, *prep*, o) where v is a verb or a normalized extended verb phrase and o is a noun or a normalized extended noun phrase and *prep* is a preposition linked with v. An extended phrase is a phrase (noun phrase or verb phrase) whose head-word is present in WordNet [48]. These phrases can be normalized either strongly or weakly by reducing the phrase to its headword or stemming and removing leading articles respectively.

their information. They then cluster these frames to remove redundant and duplicate frames/tasks. The frames the authors create can also be seen as a knowledge graph, where the nodes are extended phrases and each of the attributes of the frame are relations between other nodes. The authors use the resulting knowledge base to be able to link WikiHow tasks with YouTube videos and find that it helps to expand the search space for relevant videos.

Another work that somewhat utilizes a knowledge graph to work with procedures is "A Framework for Procedural Text Understanding"[98]. Here, the authors use entity recognition to extract concepts from each step in a procedure and link them to other concepts throughout the step and across the step by utilizing dependency parsing. What this creates is a flow graph of how every concept is used in a step, and after what concept should the step jump to the following step. Once more, this is not an explicit knowledge graph, but the flow graph can be converted into a knowledge graph. The approach that the authors take is context-less, since they are only extracting the concepts without taking into account their usage, the approach would be improved by taking into account the context. On this topic, the authors mention that: "the simple application of dependency parsing to flow graph estimation does not work well, and that it is important to focus on not only concepts but also words surrounding them". It is worth noting, however, that the approach taken in [98] gives a visual explanation as to what is occurring in a procedure (although incomplete because of the lack of context). An image of the results of their system can be seen in Figure C-5.

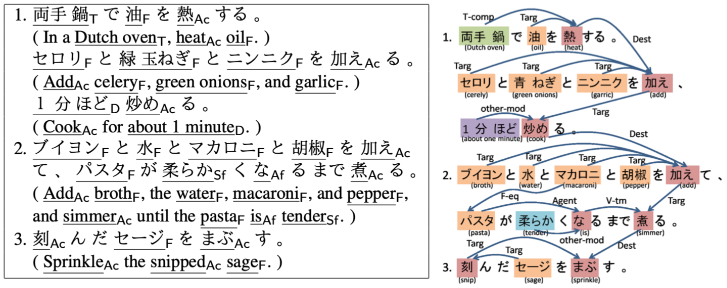

1. 両手 鍋$_{T}$ で 油$_{F}$ を 熱$_{Ac}$ する 。
(In a Dutch oven$_{T}$, heat$_{Ac}$ oil$_{F}$.)
セロリ$_{F}$ と 緑 玉ねぎ$_{F}$ と ニンニク$_{F}$ を 加え$_{Ac}$ る 。
(Add$_{Ac}$ celery$_{F}$, green onions$_{F}$, and garlic$_{F}$.)
1 分 ほど$_{D}$ 炒め$_{Ac}$ る 。
(Cook$_{Ac}$ for about 1 minute$_{D}$.)
2. ブイヨン$_{F}$ と 水$_{F}$ と マカロニ$_{F}$ と 胡椒$_{F}$ を 加え$_{Ac}$ て 、 パスタ$_{F}$ が 柔らか$_{Sf}$ く な$_{Af}$ る まで 煮$_{Ac}$ る 。
(Add$_{Ac}$ broth$_{F}$, the water$_{F}$, macaroni$_{F}$, and pepper$_{F}$, and simmer$_{Ac}$ until the pasta$_{F}$ is$_{Af}$ tender$_{Sf}$.)
3. 刻$_{Ac}$ ん だ セージ$_{F}$ を まぶ$_{Ac}$ す 。
(Sprinkle$_{Ac}$ the snipped$_{Ac}$ sage$_{F}$.)

Figure C-5: Inferred procedure flow graph from [98].

Although [98], [26], and [56] utilize knowledge graphs in different forms to understand and operate on procedures, there have been other avenues of research in procedural understanding with regards to visualizing procedures in graph-like ways. In [106] the authors test visualizing a procedure (e.g. cooking a recipe) as a workflow (see figure C-6 for an example of how this looks like). The authors find that the redesigned graph-like representation of a recipe can result in significant speed and accuracy gains across a technically inclined user base. This gives some indication that knowledge graphs can also be utilized for visualizing a procedure, which could aid in the visual explainability of the steps of a procedure, although more tests with a more traditional knowledge graph-like structure are needed.

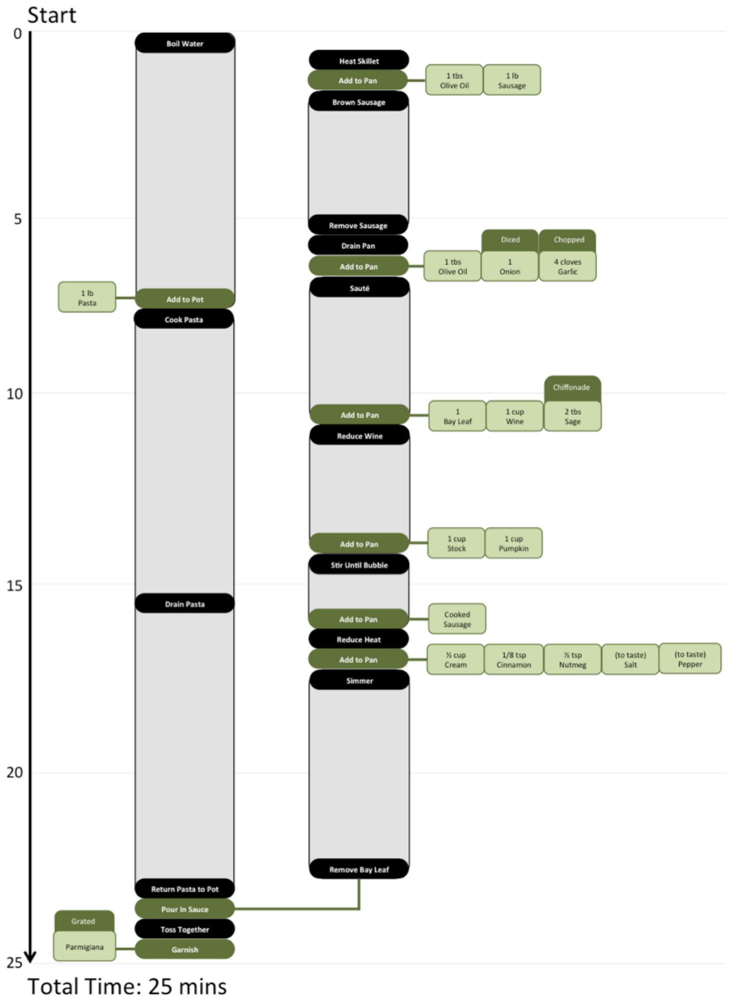


Figure C-6: Redesign of a recipe as a workflow[106]

C.2.4 Combining structured information and language models

Overall, we have seen that knowledge graphs aid in both the semantic understanding of a procedure and in the visual explanation of a procedure. We have seen that both language models and structured commonsense information (i.e. commonsense knowledge graphs) can play a key role in understanding procedures, by narrowing down the space of predictions for what happens within the steps, for generalizing to unseen entities and facts by analogy, by contextualizing and embedding entities along with their attributes from text, and by visualizing the actual procedure. There is a disconnect however between language models and structured commonsense information. We have looked at how to combine this structured information with what a language model contains [30]. We now give an overview of some approaches at combining structured information with language models. We note that since writing this work, there have been even more analysis of even more models (not only transformer based encoder systems) [193, 97, 61, 181].

Broadly speaking, there are three discrete approaches (and combinations of these) to inject structured knowledge into pre-trained transformer-based encoder language models. We call these *input focused injections*, *architecture focused injections*, and *output focused injections.* In figure C-7 We define an *input injection* as any technique that modifies the data pre-processing or the pre-transformer layer inputs that the base model uses(i.e. injecting knowledge graph triples into the training data to pre-train/fine-tune on them or combining entity embeddings into the static word embeddings that the models have). We define *architecture injections* as techniques that alter a base model's transformer layers (i.e. adding additional layers that inject in some representation). Lastly, we define an *output injection* as any techniques that either modify the output of the base models or that modify/add custom loss functions. In addition to these three basic types, there are approaches that utilize combinations of these (i.e. a system that uses both *input and output injections*), which we call *combination injections.*

For more information and examples of approaches that implement different types

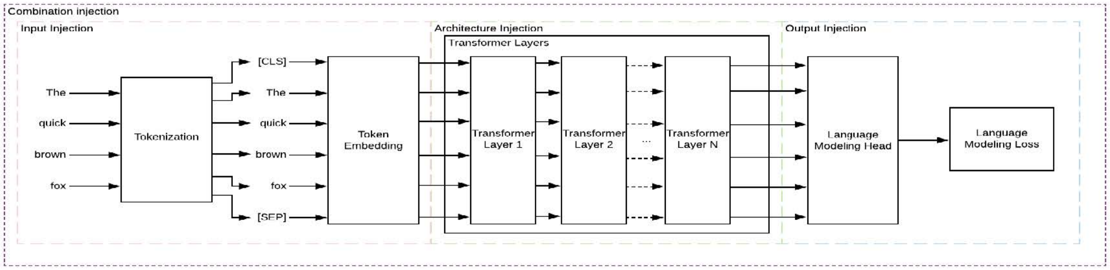


Figure C-7: Visualization of boundaries of the different categories of knowledge injections.

of injections, we defer to a survey paper that we developed [30]. Additionally, we have looked into the effects of applying knowledge graph embeddings into these injections and have found that they do indeed improve the performance of a baseline model in a downstream semantic task. As seen in [186], there is evidence that wider vocabulary in the knowledge graph in combination with more information from the entities in that vocabulary contribute to the performance increase. We address how to improve out-of-vocabulary graph embeddings and leave as future work the incorporation of graph embeddings produced from a system that we developed called RetroGAN [31]. RetroGAN is a Cycle-GAN based embedding post-specialization system that is used to learn and generalize a retrofitting mapping. Retrofitting word embeddings with a knowledge graph [47, 161, 115] means taking a vector space of word embeddings and finding a mapping that moves some of these word vectors closer together and others further apart, such that these vectors' new positions in the vector space are in better agreement with the relationships between the same words (a.k.a., concepts) in the knowledge graph [161, 115]. However, the retrofitting process can only work on concepts that are actually present in the knowledge graph or in the constraints, which means that retrofitting can only get us improved performance in semantic tasks on only the subset of the embedding vocabulary that overlaps with the knowledge graph's vocabulary.

Post-specialization[173, 72] is a solution to this problem; it is a series of techniques that try to (1) learn the mapping that retrofitting establishes and (2) generalize the mapping to the rest of the embedding vocabulary. We developed a post-specialization system called RetroGAN that builds upon the approach presented as AuxGAN [133] by extending it to have a CycleGAN-like[194] architecture rather than a regular Generative Adversarial Network (GAN[55]) architecture. By using this cyclic adversarial technique, we further constrain the outputs for unseen data in both our domains by ensuring that there is a one-to-one mapping between domains. This leads to achieving higher performance for unseen concepts in semantic comparison tasks, and in a downstream task of lexical simplification. An overview of the RetroGAN architecture can be found in Figure C-8 and more details of the performance of the system can be found in [31].

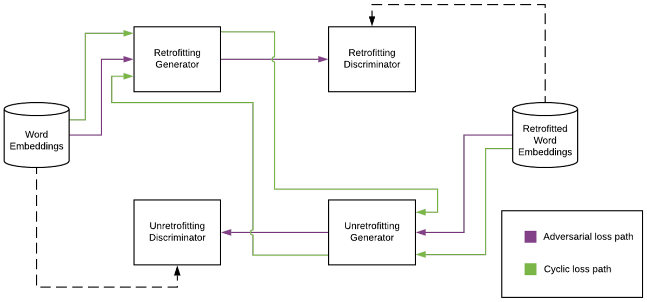


Figure C-8: RetroGAN System Architecture

The reason that we mention all of this work, is that the models that we utilize in our work (T5, BART), could, in future work, leverage many of these techniques to explicitly incorporate some of the graph structure that we generate from our contextual commonsense inference, and from broader more general knowledge graphs.

We will also note that there have been recent developments in retrieval-based methods for improving language models[129, 57, 80]. In general, retrieval methods consist of, namely, a retriever system that is in charge of finding useful information from some curated or trusted source of knowledge. This retriever is in charge of finding relevant information for a language model to inform a language model with necessary/possibly auxiliary information to perform a certain task. One popular and highly relevant work is the RAG (Retrieval Augmented Generation) model. This model utilizes a similarity based transformer retrieval system that retrieves parts of a document or from multiple documents to augment/prepend to a generation model

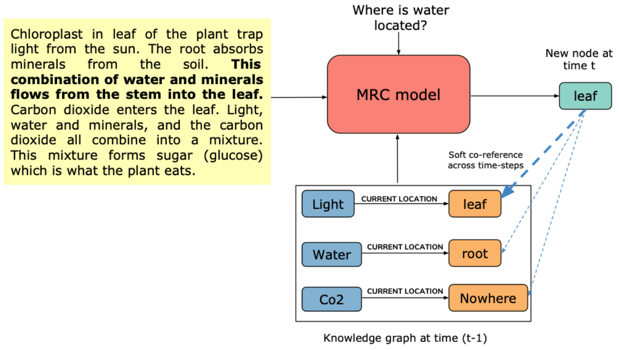
(BART) for certain tasks. The authors find that this style of system can help generate more factual and consistent summaries. We believe, although more exploration and experimentation need to be performed to show this, that this kind of system is what will be utilized in future language modelling systems.

C.2.5 Knowledge Graph Reasoning

Now, we will explore systems that have been utilized to reason on knowledge graphs. The benefit of these systems is that they are generally applicable to knowledge graphs, and could be used in future work with our graphs. The first system that we mention is called “Meandering In Networks of Entities to Reach Verisimilar Answers" (MINERVA) [34]. MINERVA is a method for efficiently searching a knowledge graph for an answer to a question, providing paths using reinforcement learning (RL) conditioned on the input question. An interesting aspect is that this system will learn an effective path to provide an answer by itself. Although not directly applicable to the problem that we have at hand, which is finding a path that represents a step, and combining these paths in order to represent a procedure, it provides a good starting point as a framework. To possibly adapt it for our purposes, we may need to append to our knowledge graph nodes that represent the start or the finish of the procedure, nodes that would serve as checkpoints in the RL system to let us know it has reached its intended destination.

A similar model falling along the lines of neuro-symbolic models is CORGI [6]. The authors develop a system that has a human in the loop in order to perform commonsense reasoning. The system takes inputs such as “if ⟨ **state holds** ⟩ then ⟨ **perform action** ⟩ because ⟨ **I want to achieve goal**⟩" and tries to find all the presumptions to support achieving the goal given that the state holds. An example that the authors use is if there is a thunderstorm in the forecast, then a person will take an umbrella with them. Their system then converses with a user until it finds that the user wants to remain dry. The authors implement a neuro-symbolic theorem prover and apply it to extract a multi-hop reasoning chain that reveals commonsense presumptions. It then prompts the user for additional information that it believes to be necessary to complete the proof. The authors noted that having the human in the loop *is essential* to their system working as some of the popular large commonsense knowledge graphs (ATOMIC, ConceptNet), were lacking some content needed for the system to reason.

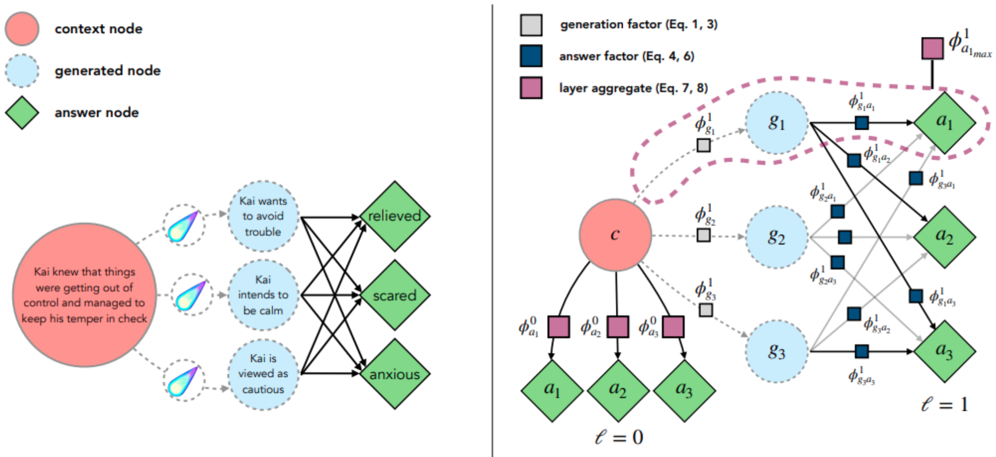
Another model that reasons with a dynamically generated knowledge graph is [35]. The authors construct, at every sentence in a procedure, a graph of entities and some of their attributes (e.g. location) to help answer questions. They utilize a machine reading comprehension model to extract new entities and update existing ones as a procedure progresses. A visualization of the process can be seen in Figure C-9. It is interesting that the model keeps adding and updating the knowledge it has as information is streamed, however there is no insight as to how multiple sources could be utilized to update a model. Our model for procedural step generation does a similar approach, however we leverage the contextual commonsense inference model to generate more commonsense assertions to help us understand what is happening in the procedure to be able to generate steps for it.



Chloroplast in leaf of the plant trap light from the sun. The root absorbs minerals from the soil. **This combination of water and minerals flows from the stem into the leaf.** Carbon dioxide enters the leaf. Light, water and minerals, and the carbon dioxide all combine into a mixture. This mixture forms sugar (glucose) which is what the plant eats.

Figure C-9: Dynamic graph construction example for [35]. When the system reads the **bolded** sentence, it adds a leaf node into its knowledge graph and updates the states of other entities in the graph.

Yet another model that constructs a knowledge graph to reason on is Dynamic Knowledge Graph Construction for Zero-shot Commonsense Question Answering[16]. The authors utilize COMET to build a dynamic reasoning chain until they arrive to a possible answer. The system that we are proposing would utilize a similar idea to this, however rather than generating plausible answers, it would be generating a path



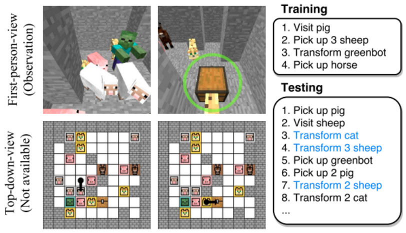
(a) COMET receives the context c and generates new commonsense inferences to construct a local graph of knowledge about the situation (Section 2).

(b) Our inference algorithms reason over the graph by aggregating commonsense paths to answer questions about the situation (Section 3).

Figure C-10: Dynamic graph construction and reasoning example for [16]. The authors build hops and perform inference on the combination of these hops to determine what is the likeliest output.

that can be translated directly into instructions for a procedure.

A more directly applicable technique is the system described by [123]. It uses a reinforcement learning agent to traverse and execute a list of instructions in a 3D environment. The system has various interesting parts. In particular, the authors handle generalization over unseen instructions, by using an analogy-like objective which encourages learning correspondences between similar sub-tasks by making analogies. The authors also present a hierarchical architecture where a meta controller learns to


First-person-view (Observation)
Top-down-view (Not available)

Training
1. Visit pig
2. Pick up 3 sheep
3. Transform greenbot
4. Pick up horse

Testing
1. Pick up pig
2. Visit sheep
3. Transform cat
4. Transform 3 sheep
5. Pick up greenbot
6. Pick up 2 pig
7. Transform 2 sheep
8. Transform 2 cat
...

Figure C-11: Visualization of instruction tasks for the system presented by [123]

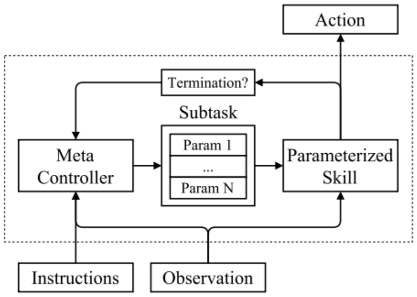


Figure C-12: Architecture for system described in [123]. The system takes in a pair of instructions and observations of the environment and produces an action.

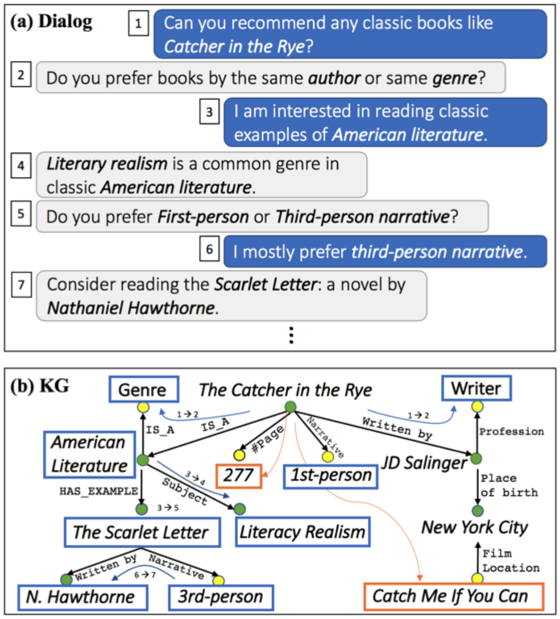
use the acquired skills for executing the instructions, the architecture handles potential interruptions by learning when to update the sub-task. The sub-tasks that the authors mention are trained by learning an optimal policy network that maps a given task embedding (e.g. a one hot vector for pickup, visit, etc.) into a set of actions. The authors make these networks that represent tasks, learn by an analogy objective in order to generalize to unseen procedures (i.e. if the sub-task is applicable to x input, and y input is similar, then the sub-task may be applicable).

C.3 Additional Work for Text Generation from Knowledge Graphs

Other work that has been done in the area of using knowledge graphs for text generation is OpenDialKG [107]. In this work, the authors extract concepts from user input text and use this to find topics/concepts that a chatbot could respond with. Although less applicable due to its primary use being to keep a user interested in an open-domain conversation, rather than a task-oriented conversation, it nonetheless provides an interesting way to utilize a knowledge graph to guide text-generation. We note that the authors explore what they call “walkable" patterns within a KG or a preferred sequence of graph traversal steps, which often leads to more engaging entities or attributes than others while pruning non-ideal (factually incorrect) KG paths.

C.3.1 Additional

We make a small note of two last topics, data formatting and the CommonGen [89] problem. Data formatting plays a role on additional pre-processing that we can do to understand procedures. It is possible to leverage the presence of ordered lists within HTML documents to determine where the steps in a procedure are. Additionally, we can leverage the numbering in them to determine the order that a procedure can be undertaken. This could possibly be leveraged in a reasoner to guide the ordering of



(a) Dialog

1 Can you recommend any classic books like *Catcher in the Rye*?

2 Do you prefer books by the same ***author*** or same ***genre***?

3 I am interested in reading classic examples of *American literature*.

4 ***Literary realism*** is a common genre in classic ***American literature***.

5 Do you prefer ***First-person*** or ***Third-person narrative***?

6 I mostly prefer *third-person narrative*.

7 Consider reading the ***Scarlet Letter***: a novel by ***Nathaniel Hawthorne***.

⋮

Figure C-13: Procedure utilized by [107] to respond with natural language text, utilizing a knowledge graph as a grounding mechanism to guide the conversation

steps.

The CommonGen [89] is a task in which a system is presented with a series of concepts, and expected to predict how these concepts can or are interacting with each other. We note that our reasoning system has to perform a similar task: interpret how elements in a knowledge graph can be traversed in such a way that we can produce steps in a procedure. Although this is similar to the CommonGen task, there is one small difference and that is that since we are pulling in additional information from the web, and that web information can serve as a ground truth to the generation problem, although we would need a way to quantify divergence from this ground truth to fine-tune the system.

C.4 Related Work for Conversational Agents

Knowing that people often prefer someone else to guide them through a procedure, there have been attempts at building agents that automate the process. There have been many successful systems that do this, although they have been mostly focused on the *presentation* of the information rather than on the *acquisition and processing* of it. Conversely, there have been systems that have focused on the *acquisition and processing*, but not on the *presentation*.

C.4.1 Classical Collaborative Agents

Around the '90s there was a series of works that were done around collaborative agents. A collaborative agent can be broadly defined as "a system that mimics the relationships that hold when two humans collaborate on a task involving a shared artifact, such as two mechanics working on a car engine together"[149]. These relatively early works did not focus too much on how they got the procedural knowledge, rather on how to convey necessary information effectively to accomplish goals or to reinforce knowledge. The assumptions being that an expert could come up with the information and that they were small enough domains that scaling would not be an issue. The following is an overview of these classical agents.

TRAINS-95 [49]

TRAINS-95 is a mixed-initiative system that utilizes voice and textual input to help a user plan some train schedules. The authors wanted to build a robust, modular, multi-modal, mixed-initiative planning assistant that would, no matter what, indicate its understanding of the user's intentions and its own attempts to further the plan. This system used a domain reasoner (the domain of train schedule planning) to be able to determine whether a proposed route was valid or could be improved and relayed this back to a user through a textual and visual interface. This collaborative system was evaluated and 73 of the 80 test sessions resulted in successful completion of the route planning task and according to the authors "this is probably a record for naive users of AI planning system". Although not explicitly a procedural guidance system, the TRAINS-95 system gave suggestions on how to perform some tasks that had an objective. The system also evaluated a user input in accordance to achieving that common goal, and it did this through both voice and text, showing that voice as a means of communication in these collaborative tasks was effective.

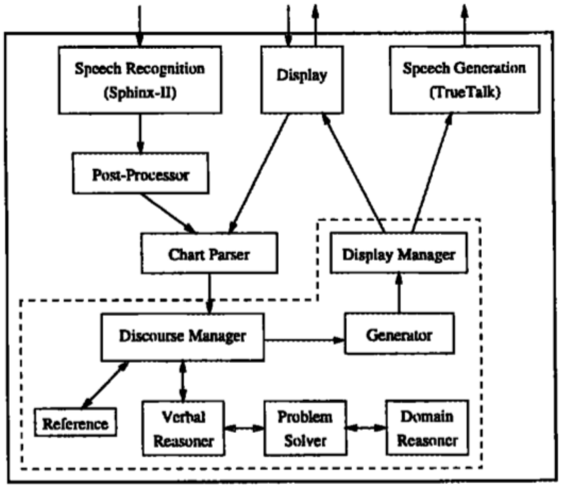


Figure C-14: Architecture for TRAINS-95 system[49]

COLLAGEN [150]

Research started converging on leveraging collaborative agents and on collaborative discourse theory (CDT). CDT is empirical and computational research about how people communicate in the context of a collaboration[151]. COLLAGEN is a system that utilizes techniques from CDT combined with a given task and its domain knowledge to create an agent that can work with a user to accomplish a common goal. The importance of this work is that the presentation is handled as long as a user can provide the domain knowledge necessary for the system to collaborate on achieving the task. One last important feature of this work is that it used dialogue as the means of communication, once more showcasing the effectiveness of conversation.

PACO [153]

Eventually, researchers incorporated the idea of not only collaborating on a task but also teaching the person how to do this task while it is being performed. Towards this end, intelligent tutoring systems (ITS) techniques were incorporated into COLLAGEN in an agent called PACO (Pedagogical Agent for COllagen). The authors

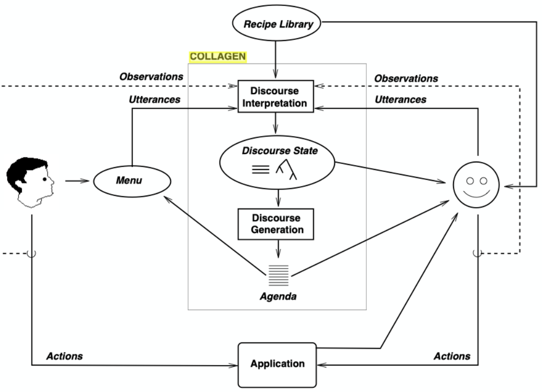


Figure C-15: Architecture for COLLAGEN based-system[150]

then map some tutorial discourse acts into a collaborative dialogue system and evaluate the use of these within the PACO system. Their results are summarized as follows: Most users commented positively on Paco's overall teaching skills, explanations, feedback, clarity of communication, and ability to understand what the user was doing, a user wanted the ability to ask "why" questions, others wanted more rationales for actions, and after their session with Paco, users felt somewhat or very confident in their ability to perform the task. [153]. This system served to showcase that it is indeed feasible to through CDT systems to actually teach people how to perform procedures (or tutorials as they refer to them), and here we see the necessity to give combinations of instruction types as we mentioned in section 2.1 (i.e. principle, example, and procedural instruction).

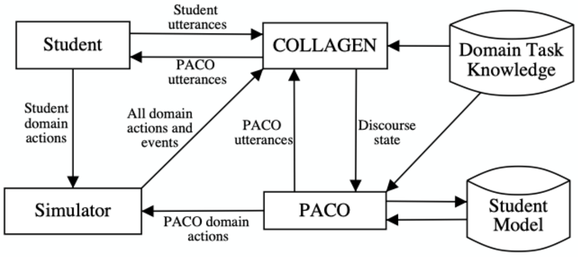


Figure C-16: Architecture for PACO[153]

Soar Training Expert for Virtual Environments (STEVE)[152]

Later on, research started to evaluate the incorporation of embodiment into collaborative agents for procedures. One such work is STEVE, a virtual reality embodied agent whose objective is to help students learn to perform physical, procedural tasks. In the work presented by [152], STEVE is evaluated on teaching a variety of naval operating procedures such as how to operate and maintain the gas turbine engines aboard naval ships. It is interesting to note that the authors mention that “Steve is not limited to this domain; he can provide instruction in a new domain when given only the appropriate declarative domain knowledge". We see that the issue of acquiring the necessary domain knowledge begin to rise in these systems. It is also interesting to note that since STEVE is embodied, he can point and look at objects of interest which helps the student in the procedure.

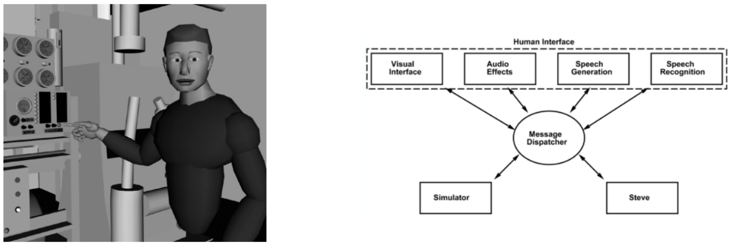


Figure C-17: STEVE embodied agent (left), architecture for STEVE (right)[153]

Summary

Altogether, these classic agents showed that it is indeed possible to build voice agents that can guide people through procedures. However, all of them assume that the knowledge necessary for them will be provided by a domain expert, which for a small domain or a handful of tasks may not be a problem, but for thousands or millions of tasks and various domains, quickly becomes an issue.

Bibliography

[1] Google i/o 2017 - applying built-in hacks of conversation to your voice ui. `https://www.youtube.com/watch?v=wuDP_eygsvs`. Accessed: 2020-04-30.

[2] Google i/o 2017 - finding the right voice interactions for your app. `https://www.youtube.com/watch?v=0PmWruLLUoE`. Accessed: 2020-04-30.

[3] Google i/o 2017 - multimodal design. `https://www.youtube.com/watch?v=fw27RFHP2tc`. Accessed: 2020-04-30.

[4] The most important design principles of voice ux. `https://www.fastcompany.com/3056701/the-most-important-design-principles-of-voice-ux`. Accessed: 2020-04-30.

[5] Aida Amini, Antoine Bosselut, Bhavana Dalvi Mishra, Yejin Choi, and Hannaneh Hajishirzi. Procedural reading comprehension with attribute-aware context flow. *arXiv preprint arXiv:2003.13878*, 2020.

[6] Forough Arabshahi, Jennifer Lee, Mikayla Gawarecki, Kathryn Mazaitis, Amos Azaria, and Tom Mitchell. Conversational neuro-symbolic commonsense reasoning. *arXiv preprint arXiv:2006.10022*, 2020.

[7] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] SK Baker, RT Santiago, J Masser, NJ Nelson, and J Turtura. The alphabetic principle: From phonological awareness to reading words. improving literacy brief. *National Center on Improving Literacy*, 2018.

[10] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine*

Translation and/or Summarization, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[11] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[12] Sumithra Bhakthavatsalam, Chloe Anastasiades, and Peter Clark. Genericskb: A knowledge base of generic statements. *arXiv preprint arXiv:2005.00660*, 2020.

[13] Priya Bhandari. Within-subjects design | explanation, approaches, examples. `https://www.scribbr.com/methodology/within-subjects-design/`, December 05 2022. Accessed: April 11, 2023.

[14] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[15] Simone Borsci, Alessio Malizia, Martin Schmettow, Frank Van Der Velde, Gunay Tariverdiyeva, Divyaa Balaji, and Alan Chamberlain. The chatbot usability scale: the design and pilot of a usability scale for interaction with ai-based conversational agents. *Personal and Ubiquitous Computing*, 26:95–119, 2022.

[16] Antoine Bosselut and Yejin Choi. Dynamic knowledge graph construction for zero-shot commonsense question answering. *arXiv preprint arXiv:1911.03876*, 2019.

[17] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*, 2019.

[18] Faeze Brahman, Alexandru Petrusca, and Snigdha Chaturvedi. Cue me in: Content-inducing approaches to interactive story generation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 588–597, Suzhou, China, December 2020. Association for Computational Linguistics.

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In

H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[21] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[22] Cédric Buche, Ronan Querrec, Pierre De Loor, and Pierre Chevaillier. Mascaret: A pedagogical multi-agent system for virtual environments for training. *International Journal of Distance Education Technologies (IJDET)*, 2(4):41–61, 2004.

[23] Yohan Chalier, Simon Razniewski, and Gerhard Weikum. Joint reasoning for multi-faceted commonsense knowledge. *arXiv preprint arXiv:2001.04170*, 2020.

[24] Xiang Chen, Xin Xie, Ningyu Zhang, Jiahuan Yan, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. Adaprompt: Adaptive prompt-based finetuning for relation extraction. *CoRR*, abs/2104.07650, 2021.

[25] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[26] Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. Distilling task knowledge from how-to communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 805–814, 2017.

[27] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[28] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[29] Michael H Cohen, Michael Harris Cohen, James P Giangola, and Jennifer Balogh. *Voice user interface design.* Addison-Wesley Professional, 2004.

[30] Pedro Colon-Hernandez, Catherine Havasi, Jason Alonso, Matthew Huggins, and Cynthia Breazeal. Combining pre-trained language models and structured knowledge. *arXiv preprint arXiv:2101.12294*, 2021.

[31] Pedro Colon-Hernandez, Yida Xin, Henry Lieberman, Catherine Havasi, Cynthia Breazeal, and Peter Chin. RetroGAN: A cyclic post-specialization system for improving out-of-knowledge and rare word representations. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2086–2095, Online, August 2021. Association for Computational Linguistics.

[32] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[33] Jeff Da and Jungo Kasai. Cracking the contextual commonsense code: Understanding commonsense reasoning aptitude of deep contextual representations. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 1–12, Hong Kong, China, November 2019. Association for Computational Linguistics.

[34] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.

[35] Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. Building dynamic knowledge graphs from text using machine reading comprehension. *arXiv preprint arXiv:1810.05682*, 2018.

[36] Joe Davison, Joshua Feldman, and Alexander M Rush. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, 2019.

[37] M. de Rijke, Harry Bunt, Jeroen Geertzen, and Elias Thijsse. Question answering: What's next? 2005.

[38] Maarten De Rijke et al. Question answering: What's next? 2005.

[39] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[41] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[42] Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. GSum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online, June 2021. Association for Computational Linguistics.

[43] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48:1–4, 2016.

[44] Elsa Eiriksdottir and Richard Catrambone. Procedural instructions, principles, and examples: How to structure instructions for procedural tasks to enhance performance, learning, and transfer. *Human factors*, 53(6):749–770, 2011.

[45] Roy Eisenstadt and Michael Elhadad. Neural micro-planning for data to text generation produces more cohesive text. In *Proceedings of the Workshop on Discourse Theories for Text Planning*, pages 6–9, 2020.

[46] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10, 2011.

[47] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

[48] Christiane Fellbaum. Wordnet. *The encyclopedia of applied linguistics*, 2012.

[49] George Ferguson, James F Allen, Bradford W Miller, et al. Trains-95: Towards a mixed-initiative planning assistant. In *AIPS*, pages 70–77, 1996.

[50] Saadia Gabriel, Chandra Bhagavatula, Vered Shwartz, Ronan Le Bras, Maxwell Forbes, and Yejin Choi. Paragraph-level commonsense transformers with recurrent memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12857–12865, May 2021.

[51] Silin Gao, Jena D Hwang, Saya Kanno, Hiromi Wakaki, Yuki Mitsufuji, and Antoine Bosselut. Comfact: A benchmark for linking contextual commonsense knowledge. *arXiv preprint arXiv:2210.12678*, 2022.

[52] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018.

[53] Carlos Gemmell, Sophie Fischer, Iain Mackie, Paul Owoicho, Federico Rossetto, and Jeff Dalton. Grillbot: A flexible conversational agent for solving complex real-world tasks. *1st Proceedings of the Alexa Prize Taskbot*, 2022.

[54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[55] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[56] Abhirut Gupta, Abhay Khosla, Gautam Singh, and Gargi Dasgupta. Mining procedures from technical support documents. *arXiv preprint arXiv:1805.09780*, 2018.

[57] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

[58] Catherine Havasi, Robyn Speer, James Pustejovsky, and Henry Lieberman. Digital intuition: Applying common sense using dimensionality reduction. *IEEE Intelligent Systems*, 24(4):24–35, 2009.

[59] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

[60] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[61] Yifan Hou, Guoji Fu, and Mrinmaya Sachan. Understanding knowledge integration in language models with graph convolutions. *arXiv preprint arXiv:2202.00964*, 2022.

[62] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[63] Xinyu Hua, Ashwin Sreevatsa, and Lu Wang. Dyploc: Dynamic planning of content using mixed language models for text generation. *arXiv preprint arXiv:2106.00791*, 2021.

[64] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022.

[65] Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*, pages 6384–6392, 2020.

[66] Filip Ilievski, Alessandro Oltramari, Kaixin Ma, Bin Zhang, Deborah L. McGuinness, and Pedro Szekely. Dimensions of commonsense knowledge. *Knowledge-Based Systems*, 229:107347, 2021.

[67] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[68] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[69] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2022.

[70] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[71] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[72] Aishwarya Kamath, Jonas Pfeiffer, Edoardo Ponti, Goran Glavaš, and Ivan Vulić. Specializing distributional vectors of all words for lexical entailment. 2019.

[73] Pei Ke, Haozhe Ji, Yu Ran, Xin Cui, Liwei Wang, Linfeng Song, Xiaoyan Zhu, and Minlie Huang. Jointgt: Graph-text joint representation learning for text generation from knowledge graphs. *arXiv preprint arXiv:2106.10502*, 2021.

[74] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[75] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[76] Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*, 2018.

[77] Douglas B. Lenat, R. V. Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd. Cyc: Toward programs with common sense. *Commun. ACM*, 33(8):30–49, August 1990.

[78] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[79] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[80] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[81] Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Yacine Jernite, Abhishek Thakur, Lewis Tunstall, Suraj Patil, Mariama Drame, Julien Chaumond, Julien Plu, Joe Davison, Simon Brandeis, Victor Sanh, Teven Le Scao, Kevin Canwen Xu, Nicolas Patry, Steven Liu, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Nathan Raw, Sylvain Lesage, Anton Lozhkov, Matthew Carrigan, Théo Matussière, Leandro von Werra, Lysandre Debut, Stas Bekman, and Clément Delangue. huggingface/datasets: 1.13.2, October 2021.

[82] Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. Large language models with controllable working memory. *arXiv preprint arXiv:2211.05110*, 2022.

[83] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. Few-shot knowledge graph-to-text generation with pretrained language models. *arXiv preprint arXiv:2106.01623*, 2021.

[84] Liang Li, Can Ma, Yinliang Yue, and Dayong Hu. Improving encoder by auxiliary supervision tasks for table-to-text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5979–5989, 2021.

[85] Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, 2016.

[86] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[87] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*, 2020.

[88] Isabelle Y Liberman, Donald Shankweiler, and Alvin M Liberman. The alphabetic principle and learning to read. 1989.

[89] Bill Yuchen Lin, Ming Shen, Yu Xing, Pei Zhou, and Xiang Ren. Commongen: A constrained text generation dataset towards generative commonsense reasoning. *arXiv preprint arXiv:1911.03705*, 2019.

[90] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[91] Yicheng Wang* Lisa Bauer* and Mohit Bansal. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the Empirical Methods in Natural Language Processing*, 2018.

[92] Hugo Liu and Push Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

[93] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021.

[94] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[95] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

[96] Ewa Luger and Abigail Sellen. " like having a really bad pa" the gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 5286–5297, 2016.

[97] Chenyang Lyu. Knowledge and pre-trained language models inside and out: a deep-dive into datasets and external knowledge. 2022.

[98] Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60, 2015.

[99] Anutosh Maitra, Shivam Garg, and Shubhashis Sengupta. Enabling interactive answering of procedural questions. In *International Conference on Applications of Natural Language to Information Systems*, pages 73–81. Springer, 2020.

[100] Andreas Marfurt and James Henderson. Sentence-level planning for especially abstractive summarization. In *Proceedings of the Third Workshop on New Frontiers in Summarization*, pages 1–14, 2021.

[101] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[102] Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. *arXiv preprint arXiv:1805.06975*, 2018.

[103] Bhavana Dalvi Mishra, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. Everything happens for a reason: Discovering the purpose of actions in procedural text. *arXiv preprint arXiv:1909.04745*, 2019.

[104] Bhavana Dalvi Mishra, Niket Tandon, and Peter Clark. Domain-targeted, high precision knowledge extraction. *Transactions of the Association for Computational Linguistics*, 5:233–246, 2017.

[105] Antonija Mitrovic, Brent Martin, Pramuditha Suraweera, Konstantin Zakharov, Nancy Milik, Jay Holland, and Nicholas McGuigan. Aspire: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19(2):155–188, 2009.

[106] Megan Monroe. Classic techniques in new domains: an alternative recipe. In *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*, pages 119–123, 2016.

[107] Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854, 2019.

[108] Robert J Moore and Raphael Arar. *Conversational UX Design: A Practitioner's Guide to the Natural Conversation Framework*. ACM, 2019.

[109] Amit Moryossef, Ido Dagan, and Yoav Goldberg. Improving quality and efficiency in plan-based neural data-to-text generation. *arXiv preprint arXiv:1909.09986*, 2019.

[110] Amit Moryossef, Yoav Goldberg, and Ido Dagan. Step-by-step: Separating planning from realization in neural data-to-text generation. *arXiv preprint arXiv:1904.03396*, 2019.

[111] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, 2016.

[112] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics.

[113] Nasrin Mostafazadeh, Aditya Kalyanpur, Lori Moon, David Buchanan, Lauren Berkowitz, Or Biran, and Jennifer Chu-Carroll. Glucose: Generalized and contextualized story explanations. *arXiv preprint arXiv:2009.07758*, 2020.

[114] Nasrin Mostafazadeh, Aditya Kalyanpur, Lori Moon, David Buchanan, Lauren Berkowitz, Or Biran, and Jennifer Chu-Carroll. GLUCOSE: GeneraLized and COntextualized story explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4569–4586, Online, November 2020. Association for Computational Linguistics.

[115] Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the association for Computational Linguistics*, 5:309–324, 2017.

[116] Dena Mujtaba and Nihar Mahapatra. Recent trends in natural language understanding for procedural knowledge. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 420–424. IEEE, 2019.

[117] Yuki Nakatani, Tomoyuki Kajiwara, and Takashi Ninomiya. Comparing bert-based reward functions for deep reinforcement learning in machine translation. In *Proceedings of the 9th Workshop on Asian Translation*, pages 37–43, 2022.

[118] Bilal Nakhal and Ronan Querrec. Cognitive embodied conversational agents in virtual learning environment. In *2017 29th International Conference on Microelectronics (ICM)*, pages 1–4. IEEE, 2017.

[119] Tuan-Phong Nguyen, Simon Razniewski, Julien Romero, and Gerhard Weikum. Refined commonsense knowledge from large-scale web contents. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[120] Tuan-Phong Nguyen, Simon Razniewski, and Gerhard Weikum. Advanced semantics for commonsense knowledge extraction. In *Proceedings of the Web Conference 2021*, WWW '21, page 2636–2647, New York, NY, USA, 2021. Association for Computing Machinery.

[121] Tuan-Phong Nguyen, Simon Razniewski, and Gerhard Weikum. Advanced semantics for commonsense knowledge extraction. In *Proceedings of the Web Conference 2021*, pages 2636–2647, 2021.

[122] Donald A Norman. *The psychology of everyday things.* Basic books, 1988.

[123] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. *arXiv preprint arXiv:1706.05064*, 2017.

[124] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[125] Cathy Pearl. *Designing voice user interfaces: principles of conversational experiences.* " O'Reilly Media, Inc.", 2016.

[126] Yixuan Pei. *Augmenting Transformers for Open Domain Procedural Text Comprehesion.* PhD thesis, Massachusetts Institute of Technology, 2021.

[127] Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49, 2018.

[128] Matthew E Peters, Mark Neumann, IV Logan, L Robert, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*, 2019.

[129] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. How context affects language models' factual predictions. *arXiv preprint arXiv:2005.04611*, 2020.

[130] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics.

[131] Malte Pietsch, Timo Möller, Bogdan Kostic, Julian Risch, Massimiliano Pippi, Mayank Jobanputra, Sara Zanzottera, Silvano Cerza, Vladimir Blagojevic, Thomas Stadelmann, Tanay Soni, and Sebastian Lee. Haystack: the end-to-end nlp framework for pragmatic builders, 11 2019.

[132] Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. Adversarial propagation and zero-shot cross-lingual transfer of word vector specialization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 282–293, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[133] Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. Adversarial propagation and zero-shot cross-lingual transfer of word vector specialization. *arXiv preprint arXiv:1809.04163*, 2018.

[134] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[135] Tambwekar Pradyumna, Dhuliawala Murtaza, J Martin Lara, Animesh Mehta, and Brent Harrison. Controllable neural story plot generation via reward shaping. In *Proc. Int. Joint Conf. Artificial Intelligence*, pages 5982–5988, 2019.

[136] Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*, 2017.

[137] Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6908–6915, 2019.

[138] Ratish Puduppully and Mirella Lapata. Data-to-text generation with macro planning. *Transactions of the Association for Computational Linguistics*, 9:510–527, 2021.

[139] PwC. Voice assistants: The next digital frontier, 2018.

[140] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[141] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[142] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[143] Mr D Murahari Reddy, Mr Sk Masthan Basha, Mr M Chinnaiahgari Hari, and Mr N Penchalaiah. Dall-e: Creating images from text. *UGC Care Group I Journal*, 8(14):71–75, 2021.

[144] Byron Reeves and Clifford Ivar Nass. *The media equation: How people treat computers, television, and new media like real people and places.* Cambridge university press, 1996.

[145] Charles Reigeluth and Byungro Lim. What is a procedural task?, Mar 1999.

[146] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[147] Danilo Ribeiro, Thomas Hinrichs, Maxwell Crouse, Kenneth Forbus, Maria Chang, and Michael Witbrock. Predicting state changes in procedural text using analogical question answering. In *7th Annual Conference on Advances in Cognitive Systems*, 2019.

[148] Leonardo FR Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*, 2020.

[149] Charles Rich and Candace L Sidner. Adding a collaborative agent to graphical user interfaces. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 21–30, 1996.

[150] Charles Rich and Candace L Sidner. Collagen: A collaboration manager for software interface agents. In *Computational Models of Mixed-Initiative Interaction*, pages 149–184. Springer, 1998.

[151] Charles Rich and Candace L Sidner. Collaborative discourse, engagement and always-on relational agents. In *2010 AAAI Fall Symposium Series*. Citeseer, 2010.

[152] Jeff Rickel and W Lewis Johnson. Task-oriented collaboration with embodied agents in virtual worlds. *Embodied conversational agents*, pages 95–122, 2000.

[153] Jeff Rickel, Neal Lesh, Charles Rich, Candace L Sidner, and Abigail Gertner. Collaborative discourse theory as a foundation for tutorial dialogue. In *International Conference on Intelligent Tutoring Systems*, pages 542–551. Springer, 2002.

[154] Julien Romero, Simon Razniewski, Koninika Pal, Jeff Z. Pan, Archit Sakhadeo, and Gerhard Weikum. Commonsense properties from query logs and question answering forums. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM ’19, page 1411–1420, New York, NY, USA, 2019. Association for Computing Machinery.

[155] Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035, 2019.

[156] Emanuel A Schegloff. Discourse as an interactional achievement: Some uses of 'uh huh'and other things that come between sentences. *Analyzing discourse: Text and talk*, 71:93, 1982.

[157] Jérémy Scheurer, Jon Ander Campos, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training language models with natural language feedback. *arXiv preprint arXiv:2204.14146*, 2022.

[158] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.

[159] Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1702–1723, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[160] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

[161] Robert Speer and Joshua Chin. An ensemble method to produce high-quality word embeddings. *arXiv preprint arXiv:1604.01692*, 2016.

[162] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *arXiv preprint arXiv:1612.03975*, 2016.

[163] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[164] Nikko Ström and Stephanie Seneff. Intelligent barge-in in conversational systems. In *INTERSPEECH*, pages 652–655, 2000.

[165] Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. Plan-then-generate: Controlled data-to-text generation via planning. *arXiv preprint arXiv:2108.13740*, 2021.

[166] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[167] Niket Tandon, Bhavana Dalvi Mishra, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. Reasoning about actions and state changes by injecting commonsense knowledge. *arXiv preprint arXiv:1808.10012*, 2018.

[168] Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard H. Hovy. A dataset for tracking entities in open domain procedural text. *ArXiv*, abs/2011.08092, 2020.

[169] Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

[170] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[171] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[172] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[173] Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. Post-specialisation: Retrofitting vectors of words unseen in lexical resources. *arXiv preprint arXiv:1805.03228*, 2018.

[174] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.

[175] Jinbo Wang, Naigang Cui, and Changzhu Wei. Optimal rocket landing guidance using convex optimization and model predictive control. *Journal of guidance, control, and dynamics*, 42(5):1078–1092, 2019.

[176] Xuan Wang, SK Ong, and Andrew Yeh-Ching Nee. Multi-modal augmented-reality assembly guidance based on bare-hand interface. *Advanced Engineering Informatics*, 30(3):406–421, 2016.

[177] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[178] Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. *arXiv preprint arXiv:2110.07178*, 2021.

[179] Giles Westerfield, Antonija Mitrovic, and Mark Billinghurst. Intelligent augmented reality training for motherboard assembly. *International Journal of Artificial Intelligence in Education*, 25(1):157–172, 2015.

[180] Bryan Michael Williams, Henry Lieberman, and Patrick Winston. A commonsense approach to story understanding. In *Thirteenth International Symposium on Commonsense Reasoning (Commonsense-2017)*, 2017.

[181] Sondre Wold. Language models and world knowledge: Injecting structured information using masked language modeling and adapters. Master's thesis, 2022.

[182] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[183] Sen Yang, Yang Liu, Dawei Feng, and Dongsheng Li. Text generation from data with dynamic planning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:26–34, 2021.

[184] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[185] Go Yasui, Yoshimasa Tsuruoka, and Masaaki Nagata. Using semantic similarity as reward for reinforcement learning in sentence generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 400–406, 2019.

[186] Claire Yin and Henry Lieberman. Getting help from the neighborhood: Local semantic averaging for commonsense inference in language models. *ACS 2022*, 2022.

[187] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[188] Li Zhang. Reasoning about procedures with natural language processing: A tutorial. *arXiv preprint arXiv:2205.07455*, 2022.

[189] Li Zhang, Qing Lyu, and Chris Callison-Burch. Reasoning about goals, steps, and temporal ordering with wikihow. *arXiv preprint arXiv:2009.07690*, 2020.

[190] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[191] Zhihan Zhang, Xiubo Geng, Tao Qin, Yunfang Wu, and Daxin Jiang. Knowledge-aware procedural text understanding with multi-stage training. In *Proceedings of the Web Conference 2021*, pages 3512–3523, 2021.

[192] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[193] Chaoqi Zhen, Yanlei Shang, Xiangyu Liu, Yifei Li, Yong Chen, and Dell Zhang. A survey on knowledge-enhanced pre-trained language models. *arXiv preprint arXiv:2212.13428*, 2022.

[194] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Veni, vidi, vici -Julius Caesar