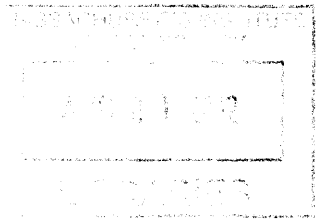
Unsupervised Discovery of Human Behavior and Dialogue Patterns in Data from an Online Game



by

Tynan S. Smith

S.B., Massachusetts Institute of Technology, 2010

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirement for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

August 2011

Author_____
Department of Electrical Engineering and Computer Science
August 1, 2011

Certified by_____
Deb Roy, Associate Professor of Media Arts and Sciences
Thesis Supervisor

Certified by_____
Jeff Orkin, Ph.D. Candidate
Thesis Co-Supervisor

Accepted by_____
Dr. Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Unsupervised Discovery of Human Behavior and Dialogue Patterns in Data from an Online Game

by

Tynan S. Smith

Submitted to the

Department of Electrical Engineering and Computer Science

August, 2011

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

A content authoring bottleneck in AI, coupled with improving technology, has lead to increasing efforts in using large datasets to power AI systems directly. This idea is being used to create AI agents in video games, using logs of human-played games as the dataset. This new approach to AI brings its own challenges, particularly the need to annotate the datasets used. This thesis explores annotating the behavior in human-played games automatically, namely: how can we generate a list of events, with examples, describing the behavior in thousands of games. First dialogue is clustered semantically to simplify the game logs. Next, sequential pattern mining is used to find action-dialogue sequences that correspond to higher-level events. Finally, these sequences are grouped according to their event. The system can not yet replace human annotation, but the results are promising and can already help to significantly reduce the amount of human effort needed.

Thesis Supervisor: Deb Roy

Title: Associate Professor of Media Arts and Sciences

Acknowledgments

I would like to give special thanks to Jeff Orkin, who I have greatly enjoyed working with for the past five years. He was a great help throughout my thesis, as my go-to person whenever I encountered a new challenge and needed someone to bounce ideas off or get inspiration from. He was also incredible in getting a grant that made it possible for me to work with him as a full-time research assistant during my last year. I would also like to thank Deb Roy, who, although he was busy starting a company while on sabbatical this year, agreed to be my faculty supervisor and always had insightful ideas. More thanks go to Hilke Reckman, who was a swell office mate, although we often weren't in the office at the same time.

I would also like to thank my family and friends, who helped keep me optimistic and motivated when my thesis was threatening to overwhelm me. I would especially like to thank my parents, who provided fresh sets of eyes to help me revise the written thesis. I would also like to thank my sister, who I dragged into traveling with me to give me something to look forward to after my thesis was done. I would also like to thank Hanh Pham who was always providing encouragement and someone to hang out when I needed a break. Lastly I would like to thank Harley Zhang, who wrote a thesis so that mine could blow it out of the water, which it does.

This work and my research work with Jeff Orkin was supported by a grant from the Singapore-MIT GAMBIT Game Lab for "Collective AI for Social Role-Playing Agents."

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the holy grails of artificial intelligence is creating computer controlled agents that can interact with humans well enough to be indistinguishable from a human controlled agent. Indeed, Alan Turing, who was instrumental in the development of computer science, proposed attempting to distinguish between a human and a computer interacting with each other through dialogue as a measurement of the "intelligence" of computers [1, 2]. This idea became known as the "Turing test," and it has become one of the standard methods used to evaluate the performance of AI dialogue systems, being used annually to judge the winner of the Loebner chatter-bot competition [3]. Creating good dialogue systems is one of the main focuses in both industry and academia for creating AI agents to interact with humans. In addition to chatter-bots, academic work has included robotic portraits [2], training systems [4], and interactive games [5]. Industry work has included a lot of task-oriented systems, like airline booking bots [6].

One of the greatest challenges with developing AI agents that interact with humans is capturing the breadth of human behavior. It is extremely difficult to make agents robust enough to deal with any possible situation they might encounter while interacting with a human. Some AI developers handle this dilemma by limiting the ways humans can interact with an AI. This is often seen in video games where the only dialogue with AI characters typically consists of selecting one of a given set of things to say (see figure 1.1), and actions like bartering or attacking are often restricted to certain situations or certain characters. Another approach used in many commercial phone systems and chat-bots is to have a catch-all response to unrecognized input. For example, many of the early chat-bots like ELIZA [7] and PARRY [8], consisted primarily of catch-all pattern-matching responses that were designed to give the appearance of a particular personality that would hide the problems with the natural language system [2].

However, if the desire is to have a system that can respond realistically and appropriately to a wide-array of input, a large database of behavior is needed. If an agent designer is scripting the agent's behavior by hand they would have

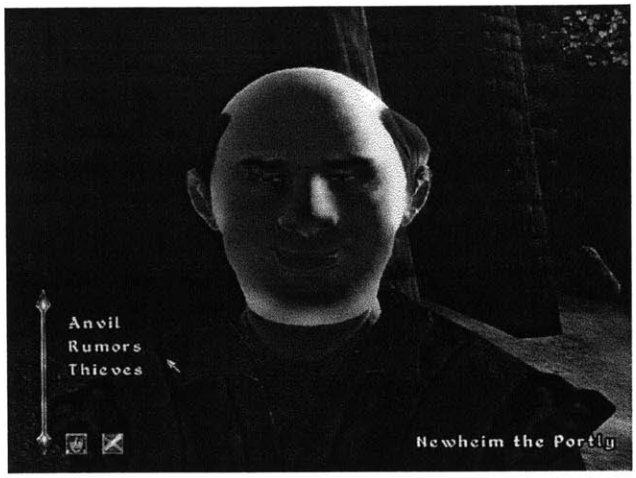


Figure 1.1: A screenshot of the game *Oblivion* showing an example of the limited dialogue allowed with AI characters in most video games. The dialogue lines available to the user are presented as a short list to be selected from.

to enumerate and script responses to all the scenarios the agent might encounter. In any but the most limited forms of interaction this approach quickly becomes difficult to scale. For example, modern pattern-matching chatter-bots using AIML have tens of thousands of pattern and response pairs that must be crafted by hand [2]. Another example is the dramatic game *Façade*, which allows the player to interact with two AI characters using typed natural language (see figure 1.2). It took the two researchers who developed the game five years to hand-script the behavior of the characters in this 15 minute game [5]. This content authoring bottleneck is a key problem that has prevented AI from advancing as far as we would like. As Roger Schank, an influential contributor to artificial intelligence in the 70s and 80s, said:

> "Not as much has happened in AI as one might have hoped. ... The major reason for this is really one of content rather than structure. ... we simply were not able to even begin to acquire the content of a [human] memory." — Roger Schank [9]

Due to this bottleneck, AI researchers and developers have been moving in a new direction, in which behavior and knowledge is not hand scripted. Instead behaviors are learned in an automated or semi-automated fashion from large human-generated datasets (such as recorded interactions, or Internet content), reinforcement learning, or some other method. Many approaches in this area have focused on using large datasets to learn a smaller simplified model that is easier to use. For example, Lane and Brodley build user models from command sequences for a computer security system [10]; Barzilay and Lapata learn a model of text coherence based on entities from a dataset of coherent texts [11] etc. In fact, the entire AI sub-field of machine learning is focused on learning smaller models from training data sets.

The problem with these models is that they contain less information than the dataset they were built from. This makes them more efficient and focused, but it also makes them less useful for applications involving human-machine

Figure 1.2: A screenshot of the drama game *Façade*, which allows players to interact with AI characters through natural language.

interaction where we want to be able to draw on all possible experience. With our ever increasing processing power, memory size and disk space, and the ability to run programs on large clusters of servers, new approaches have been made possible. Now instead of learning a small model from a large dataset, we can modify the dataset itself in such a way that it can be used directly as the source of knowledge or behavior. For example, IBM's Watson, which proved very capable at Jeopardy, is powered by a large database constructed from Wikipedia and other on-line sources [12, 13].

Jeff Orkin, a PhD candidate at the Media Lab of MIT who I have been working closely with, has been pursuing a similar technique for video game AI. He uses an entire database of recorded human behavior that has been annotated and is accessible by the AI system during runtime [14]. Orkin is developing a new approach to authoring AI for video games and training systems by capturing behavior from humans rather than hand-scripting patterns of behavior [14]. Some similar work has been done with case-based reasoning for real-time strategy games [15], but Orkin's work differs in that it works with social behaviors and a seamless mix of actions and dailogue, with tens of thousands of possible actions and utterances. Orkin is currently working with data from a two person restaurant simulation he made called The Restaurant Game [16].

In the Restaurant Game, human players control either a customer or a waitress and are instructed to act out a normal restaurant scenario. Although the game engine allows for a wide variety of behavior (everything from eating the flowers on the table to stacking cheesecakes in order to climb onto the roof), typical patterns of behavior emerge. Orkin originally implemented a fairly simplistic AI that matched patterns from the game it was experiencing during run-time to un-annotated games in its database [14]. However, this approach was unable to appropriately capture the overall structure of a game (e.g. the customer might keep ordering food and eating forever) or dependencies between events in the game (e.g. the customer might order steak but receive salmon). To address these issues, Orkin

is developing an AI system that uses an annotated version of the game log database. Currently, he has developed tools that allow actions and dialogue lines (which I will collectively refer to as "game-actions") to be grouped into "events" such as ordering food, or paying the bill [14]. Work is currently progressing on further annotations of cause and effect relationships, such as the relationship between ordering a salmon and the waitress bringing a salmon.

Annotated datasets, including the one being built from the Restaurant Game, provide additional information to an AI system about what the data means that is not easy to automatically observe in the data. However, although it is not typically hard to find or generate a database of human behavior or dialogue, it can be difficult and time-consuming to annotate the dataset with enough information to make it useful. Orkin's current approach to overcoming this problem has been to create easy-to-use annotation tools that can be crowd-sourced in order to quickly annotate large datasets [17]. Even with these tools, it takes an expert who is familiar with the data to generate the possible annotations. This can be a challenge itself, especially in a domain more complicated than a restaurant scenario. The difficulties of manually annotating databases of human speech and behavior provided the motivation for this thesis. Given the corpus of thousands of game logs from pairs of humans playing the game, how can we automatically generate a list of events describing the behavior discovered in the logs, and a set of example sequences for each event.

In this thesis I use the game play logs from The Restaurant Game to complete as much of the process of event annotation as possible in a completely automated, unsupervised manner. As mentioned previously, event annotation in the context of the restaurant game logs involves grouping together actions (e.g. pickup glass, give menu, eat cake) and dialogue lines into higher-level events. The sequences of game-actions that compose different events may be interleaved within game logs (see figure 2.1). Furthermore, a single event, such as the customer getting seated, may have many different forms in different logs.

Therefore, correct event annotation has two parts. First, within a single log the actions and dialogue should be grouped correctly into their respective events. Second, corresponding event-sequences in different game logs should be recognized as belonging to the same event. The ideal, automatic, unsupervised system would be able to correctly annotate a set of game logs given nothing but the game logs themselves. The automated event-sequence discovery system I have developed combines algorithms and concepts from many different areas of natural language processing, machine learning, artificial intelligence and other areas of computer science in a novel new way. The process used by the system can be summarized as:

1. Process the game logs to turn them into a sequence of actions and dialogue lines.

2. Cluster words semantically and functionally based on context and surface similarity.

3. Cluster dialogue lines and actions based on context and surface similarity.

4. Rewrite the logs as dialogue and action cluster sequences.
5. Mine the logs for frequent patterns with certain characteristics that are likely event-sequences.
6. Filter the candidate event-sequences based on certain properties to remove most non-event-sequences.
7. Find the optimal labeling of likely event-sequences in each game log.
8. Cluster the event-sequences by the event they correspond to.
9. Evaluate the performance of the system relative to a golden, manual labelling of the logs.

Although my system can not perfectly annotate the game logs, it does make the annotation process much easier. It is able to automatically partially annotate the events in Restaurant Game logs (typically about a third of each log) and generate a reasonable set of about 50-60 event clusters. The partial labeling assists human annotators by taking care of many of the most typical event-sequences, requiring them to only annotate more unusual sequences and correct mistakes the system made. The event cluster output is an organized collection of behavior sequences that appear in the game logs, which greatly simplifies the process of creating the set of event labels that an expert must perform. It also produces typical example sequences for each event, which can be used to train human annotators.

Due to the complexity of the overall system, most of the remaining chapters will be organized by opening with an overview and then discussing the system in terms of each of its components. Chapter 2 presents related work and relevant research for the whole system as well as individual parts and discusses the theory behind the design in more detail. Chapter 3 discusses the implementation of the system and each of its components in great detail. Chapter 4 examines the results and performance of the system overall as well as several of its key components and investigates what works well and what doesn't. Lastly, chapter 5 concludes, elaborating on the potential usefulness of the system, the contributions it makes to several areas of research, and how it might be improved in future work.

Chapter 2

Background, Related Work and Theory

This chapter discusses some of the multitude of related work that has been done in a variety of fields as well as the theory behind my approach and choice of algorithms. The first section details previous and on-going work that Jeff and I have done related to the overarching video game AI project that drove this thesis. The second section discusses some of the research that has similar objectives to the whole system developed in this thesis. The remaining sections talk about the components of the system and how the algorithms in each were chosen from a variety of options developed by other researchers. For more information about the specific implementation of each of the components, see chapter 3.

2.1 The Restaurant Game Project

This thesis is inspired by a larger video game AI research project that I have been working on and was started by Jeff Orkin, a PhD candidate at the MIT Media Lab. The project is developing the idea of collective AI. Jeff is developing a system to control AI characters which uses a database of annotated examples of human-played games to select appropriate dialogue and actions [14].

The motivation behind the project is multifaceted. First it eliminates the need for expert AI programmers to manually script all the possible behaviors an AI could exhibit. This is useful because it removes the need for experts and because when manually scripting behavior it is extremely difficult even for experts to create a robust, interesting AI capable of interacting with other players in a variety of interesting ways. Second, since the recorded games are being annotated and used directly, rather than a behavior model learned and extracted from them, the AI can recreate unusual behavior as well as typical behavior when the situation is appropriate. Furthermore, it is relatively easy to gather data because

of the massive number of hours of video games being played on a daily basis. For example, as of February 2010, the collective play time of all players in World of Warcraft was 5.93 million years [18]. Another motivation is the idea of running the AI on the cloud. AI servers could control characters for people around the world, and to some extent characters created using a dataset from one game could be placed into another game.

The current game we have been working with is called The Restaurant Game [19]. It is a simple two-player simulation of a restaurant experience, from the customer entering and sitting down to paying and leaving. One player controls the waitress and the other controls the customer. Each is instructed to act out a typical restaurant scenario, but the game engine itself allows for a wide variety of behavior, from eating the flowers on the table to stacking cheesecakes in order to climb onto the roof of the restaurant. The dialogue that takes place is typed by each player and is completely unrestricted. Since the game's release in 2007, we have gathered over 10,000 recorded games played by over 15,000 unique people.

The current iteration of the AI is collective in two ways: first it records the actions and dialogue of humans playing a game [16]; and second it uses an on-line tool to crowd-source annotation of the recorded game play logs [17]. It is the annotation process that my thesis is related to. One of the types of annotations that are applied to logs are event-sequence labels. This groups the game-actions by the events that they are a part of. For example, the following sequence in a log would be annotated with the "Waitress cleans table" event label.

1. Waitress says "Ill take this for you."
2. Waitress picks up dirty dish.
3. Waitress puts dirty dish on counter.
4. Waitress cleans dirty dish.

Jeff has developed an easy to use web-based tool that allows anyone to perform this annotation on the game logs. He has also shown that un-trained people, who do not have prior knowledge or experience with the system and he has never met, can use it almost as well as himself [17]. A screenshot of the tool is shown in figure 2.1. This tool could be used with existing crowd-sourcing technology, such as Amazon Mechanical Turk, to quickly and cheaply annotate a large number of game logs.

This process works well in the case of The Restaurant Game, but it might be more challenging for a more complicated, or unusual game. Almost everyone knows what to expect from a typical restaurant scenario so it is easy for them to identify the events in a log when given a set of labels. However, in our newest game, Improviso [20], which is an open-ended science-fiction improvisational acting game, it could be much more difficult to identify event instances in

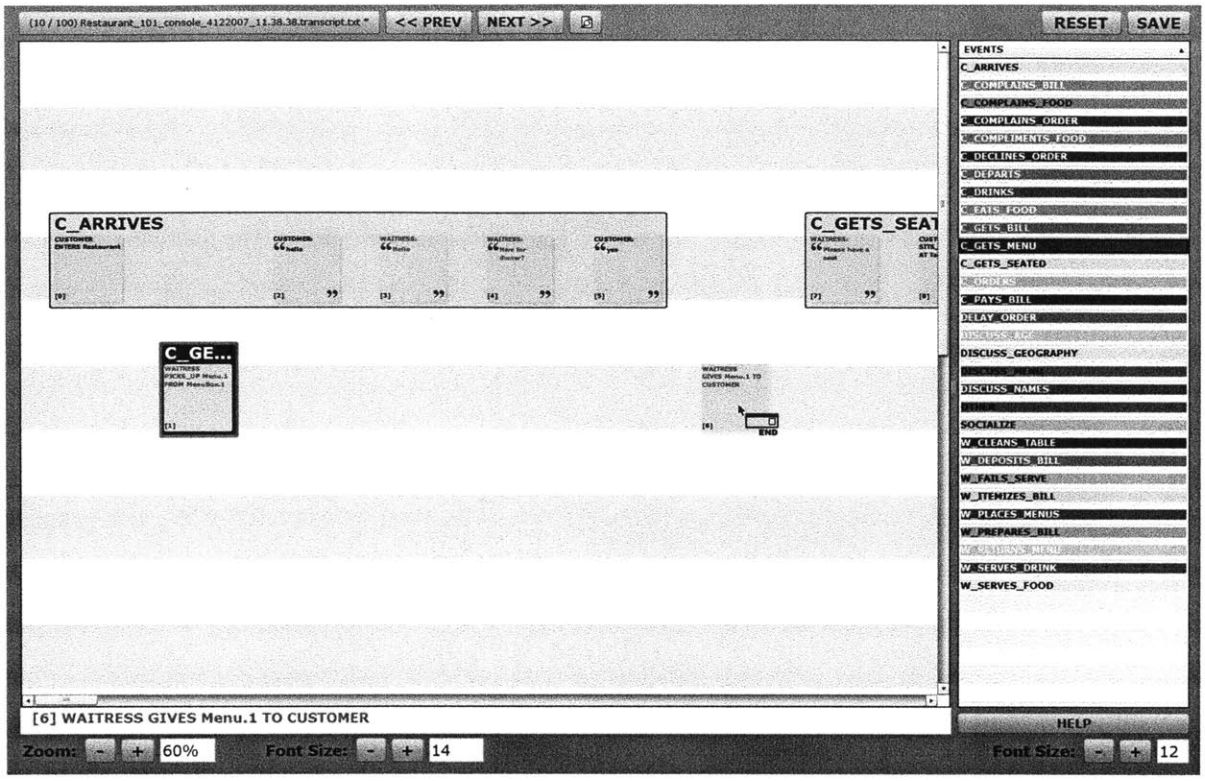


Figure 2.1: This shows the interface of the web-based tool developed to allow anyone to annotate event-sequences in game logs. The user is currently putting the bottom two actions in a "CUSTOMER_GETS_MENU" event-sequence.

logs, let alone generate the set of event labels to begin with. Even in the case of the restaurant situation, it is not a trivial matter to generate the set of events. It took Jeff several iterations of creating a set and attempting to label logs with it before arriving at the final set he is currently using.

The challenges of manual event annotation are what lead me to develop the goal of this thesis. The original goal was to create a system that could entirely automatically, find and label event-sequences in the logs and cluster the sequences into events. However, it is very difficult for an automatic system to catch all the event-sequences accurately due to the extreme variation of human-behavior and dialogue. The goal then became to develop a system that could help with the event labeling process by finding and clustering the most common event-sequences. This can help when choosing a set of event labels by using the extracted clusters as reference. It can also help train people to use the annotation system on the dataset by providing examples of each label. Lastly the system can partially label the logs so that hand-annotators only have to correct its mistakes and label rare event-sequences. This could significantly reduce the time and cost associated with annotating the logs, and more importantly help make the process more data-driven so that an expert does not have to spend a lot of time pouring over the logs to come up with a set of event labels.

2.2 Related Work

Although this exact problem of mining, clustering and labeling event-sequences is new, it is similar to work going on in many different fields. Some of the most related research has been in the areas of sequential pattern mining and learning, dialogue systems, and a variety unsupervised natural language processing efforts. The following subsections describe the work that has been done in each of these areas and the similarities and differences they share with this thesis project. The sections after this discuss the prior research efforts that influenced the various components of this system.

2.2.1 Sequential Pattern Mining and Learning

Sequence pattern mining is one of the key stages in my event-sequence discovery system so it is natural that a lot of the other research involving sequence mining and learning is related to my work. There has been a lot of similar work done with mining patterns of human behavior associated with the development of sequential pattern mining algorithms. However, most of the work deals with the actions of a single person, and a lot of it is only concerned with contiguous behavior sequences. One of the main applications of sequential pattern mining, web-log mining, attempts to find patterns in the series of actions a single user performs on a website [21]. The goal of web-log mining is typically focused on user modeling [21], rather than the focus on event-sequence discovery of this thesis.

Another similar field is sequence learning and classification, in which a sequence (typically representing human actions of some form) is assigned a class based on previous examples a system has learned from. One of the main applications of this is computer security, in which the commands entered by a user are classified as a anomalous or normal in order to detect intruders or a user behaving inappropriately [10]. This is often done by building a database of sequences representing normal behavior for each user and comparing current actions to the database in real-time [10, 22]. Lau has explored another application of sequence-learning approaches, developing intelligent user interfaces [23]. One challenge she has noted is identifying the task a user is performing based on the sequence of actions she has performed. Although the sequence-learning ideas of comparing, clustering and classifying sequences resembles my event-sequence discovery system, these approaches typically only look at a sequence as a whole, without trying to break it up into individual events.

One last similar area of sequence research is sequence prediction. In sequence prediction the goal is to find statistical regularities in some set of training examples in order to predict the next element in a novel sequence. This is applicable to many problems in computer science, such as text compression, dynamic optimization and predictive caching [24], as well as intelligent user interface design [23]. Sequence prediction is probably the least similar to event-sequence

discovery, but the notion of finding statistical patterns in sequences beyond simple pattern mining resembles some of the additional steps involved in filtering and clustering event-sequences in my system. Also, many of the algorithms used in sequence prediction extract interesting models from the observed sequences such as HMMs [23], which I explored as possible ways to find and model event-sequences.

Much of the discrete sequences related research is very similar to my project. I did a lot of background research in these areas, gathering ideas for the system overall and ultimately selecting a pattern mining algorithm to be the basis of my candidate event-sequence mining component. Section 2.3 discusses the sequence mining component of the event discovery system in more detail, discussing related research and the research that influenced it. Section 3.6 describes the specific implementation used and how it was modified from the original PLWAP sequential pattern mining algorithm.

2.2.2 Dialogue Systems

Two of the main differences between my thesis project and most discrete sequence mining are: discrete sequence mining usually focuses on the actions of a single person rather than multiple interacting people; and discrete sequence mining typically deals with a small set of possible elements, rather than the diversity provided by allowing natural language and open-ended behaviors. Dialogue system research deals with both of these things. Dialogue systems are many and varied, from simple chat bots like ELIZA to the complex speech systems such as airline booking systems[6]. The goal of dialogue systems is for a computer to interact with a human through natural language in a realistic fashion, often in order to accomplish some specific task.

One of the areas of dialogue research most relevant to event-sequence discovery is the processing of databases of human-human dialogues. For example, Chotimongkol and Rudnicky have shown promising results for applying unsupervised learning algorithms to the problem of acquiring domain-specific dialogue information from human-human interactions [25]. Their goal is identical to mine, "reducing human annotation effort," [25], and their problem very similar. They are trying to annotate human-human task-oriented dialogues, such as a customer reserving a flight and a hotel from a travel agent, with their own form-based dialogue structure representation information [25]. Their dialogue structure representation identifies subtasks in the dialogue and fills in fields in corresponding forms from the information in the dialogue, see figure 2.2.

Many of the steps in their process are very similar to stages in my own system: they cluster words based on contextual similarity much like in my system (see sections 2.4.1 and 3.3); they segment the dialogue into subtasks, similar to my labeling logs with event-sequences (see sections 2.5 and 3.8); and they cluster the subtask segments to group instances of the same subtask together (see sections 2.4.3 and 3.9).

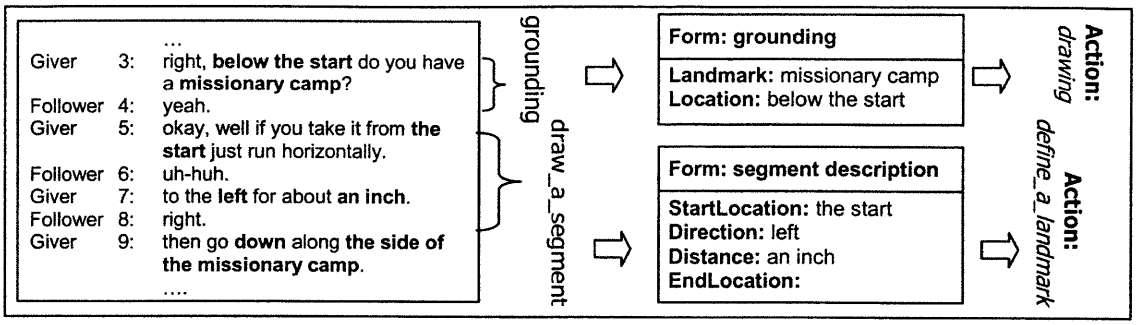


Figure 2.2: "An example of a dialogue in the map reading domain and its corresponding [Chotimongkol and Rudnicky] form-based representation." This figure was taken from their original paper [25]

One of the main differences between their research and my event-sequence discovery system is that they are dealing only with dialogue, not actions at all. Also since the dialogues are fairly short and task-oriented, they don't have to deal with a lot of overlap between subtasks so their subtask segmentation only has to identify boundaries between the subtasks rather than identify a possibly interrupted sequence of action and dialogue lines corresponding to a subtask as in my case. Furthermore, they assign every line of the dialogue to a subtask, unlike my system which leaves many actions unlabeled. This is a useful feature for my system because in the domain of an open-ended video game, many of the actions and dialogue lines are not part of any typical restaurant event, people like to do silly things. Their representation of subtasks is a form with concept fields that are filled with words from the dialogue, whereas I do not create a separate representation beyond clustering the dialogue lines and actions.

Gandhe and Traum have also done some relevant dialogue research, which is similarly motivated by creating virtual characters capable of interacting with humans through natural language [4]. Their goal is to build a statistical dialogue model from an un-annotated human-human dialogue corpus to avoid the massive amount of expert effort required to create typical rule-based dialogue systems. This research is very similar to the overall AI project motivating my thesis. However, there are some differences in how we handle the un-annotated corpus. The main difference is that my goal is to automatically annotate the corpus with useful event information that can then help the AI system Jeff is developing. However, their goal is to develop a chat bot that can use the un-annotated corpus directly using context and skip the annotation step altogether. We found this did not work well in our domain because there was no concept of the complete structure of a restaurant scenario so two bots could wind up repeatedly ordering food forever or leaving before paying etc. We are dealing with this by adding in event-sequence annotations, they are dealing with it by segmenting dialogue based on whether or not expert-chosen key concepts have been mentioned. Their approach is simpler and makes it easy to automatically segment all the dialogues once the key concepts have been chosen.

Olney describes another unsupervised dialogue system similar in nature to collective AI [2]. His approach is to use a database of dialogues to respond to a human user in real-time by using the human's utterance to find the most

appropriate response in the database. His scoring of appropriateness is more complex than the context-based one used by Gandhe and Traum, but it still has no notion of history and what has been talked about previously.

Dialogue systems based on human-human corpora are similar to our collective AI project, so a lot of the techniques they use for annotating or otherwise processing the corpus are similar to the techniques used in my event-sequence discovery system. They have similar challenges of trying to select responses that are appropriate based on the history of the interaction so far. However, dialogue systems do not have to deal with actions, and they're datasets tend to be more cohesive so they don't have to deal with interleaved events nearly as much.

2.2.3 Unsupervised Natural Language Processing Research

In addition to dialogue systems, other areas of natural language processing have produced research on unsupervised techniques related to event-sequence discovery, only some of which I mention here. Barzilay and Lapata developed an effective unsupervised coherence assessment algorithm based on the pattern of entity usage within a discourse [11]. Valid event-sequences must be coherent so the process of event-sequence discovery is largely composed of finding coherent sequences of dialogue and actions. Barzilay and Lapata assess coherence of new text by automatically extracting a model from texts known to be coherent, whereas I attempt to find the coherent sub-sequences in logs assumed to be predominantly coherent. In addition, their model is based on the transitions in the grammatical role entities fill from one sentence to the next, while my system evaluates coherence using the probability of particular clusters of actions and dialogue lines appearing sequentially. Their approach would definitely be worth exploring as an additional piece of sequence mining and filtering in future work.

Chambers and Jurafsky describe an interesting unsupervised system for learning narrative schemas, "coherent sequences or sets of events ... whose arguments are filled with participant semantics roles defined over words" [26]. Their system automatically discovers sets of related events and information about the possible values of their arguments. Similar to Barzilay and Lapata, event chain coherence is evaluated in terms of the arguments they share. However, in Chambers and Jurafsky event chains are only partially ordered and need not be continuous in the data, which is somewhat similar to my model of event-sequences.

In both of the above unsupervised NLP research, entities play an important role in determining coherence. In my event-sequence discovery system, entities are ignored in favor of clustering dialogue lines semantically. In future work it would be interesting to examine the possibility of using entities to find coherent sequences, rather than statistics. Reckman has already shown that the Restaurant Game data can be used to associate objects in the game with the names of those objects based on co-occurrence in dialogue lines [27].

2.3 Sequence Mining

One of the key components of the event-sequence discovery system is the candidate event-sequence mining stage. I explored a lot of algorithms in two areas of computer science before deciding on the modified PLWAP sequential pattern mining algorithm as described in section 3.6. Dynamic Bayesian Network (DBN) learning has the advantage that it produces fewer, more accurate candidate sequences because it tries to learn the cause and effect relationships between dialogue lines and actions. It produced promising results on simplified cases, but did not scale well to the full-sized problem. Sequential pattern mining techniques are efficient and allow for more direct control over the types of sequences found, but this is also a detriment as it is not easy to determine the best value of the controlling parameters. More information about several algorithms from each field and how they can be applied to event-sequence discovery are given in the following two subsections. Further exploration of DBNs would be a good direction for possible future work.

2.3.1 Sequential Pattern Mining Algorithms

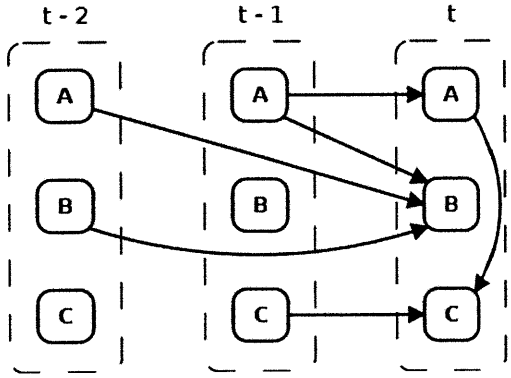
One of the most similar areas of research to this project is sequential pattern mining, which is the generalized problem of finding recurring subsequences in a database of sequences. Sequential pattern mining has many different algorithms and variations depending on the needs of a particular problem. They are an important first step in many different data-mining and processing applications, especially web-log analysis [21]. Sequential pattern mining algorithms are simplistic in the sequences they find, typically only selecting sequences based on their frequency of occurrence rather than any meaning they might have. Some sequential pattern mining algorithms have been adapted to find only maximal (not the subsequence of any other frequent sequence) or closed (not the subsequence of any sequence occurring with the same frequency) frequent sequences [21].

Some of the currently popular general-purpose sequential pattern mining algorithms include PrefixSpan [28], PLWAP [29] and SPADE [30]. Each one has its own advantages and disadvantages depending on the problem they are to be applied to. Mabroukeh gives a good summarization of the properties of these three algorithms and others [21]. There are also many specialized sequential pattern mining algorithms that have been used for various problems in NLP and elsewhere, such as DIMASP-C and DIMASP-D [31]. The problem I found with these specialized algorithms was, that although they were very efficient, they were also very specialized and difficult to adapt to my particular problem. In particular, the DIMAPS algorithms were only designed to find contiguous frequent sequences. Ultimately I settled on PLWAP, because it was one of the fastest algorithms [21] and because it was relatively simple and easy to modify for my purposes.

2.3.2 Dynamic Bayesian Networks

Another set of algorithms that can be used to mine sequences from data focus on finding sub-sequences based on cause and effect relationships rather than simple frequencies of occurrence. Dynamic Bayesian Networks (DBNs) can be used to find the cause and effect dependencies between the set of action and dialogue lines the sequence database contains.

DBNs are directed networks of variables in which an arrow from variable A to variable B means that the value of variable B is dependent on the value of variable A. They are "dynamic" because they are based on temporal sequences of actions, with a copy of all the variable nodes for each time-slice represented. For example, there might be three variables A, B and C, so we would have a node for each variable at the present time, a node for each variable at time t-1 etc. for however many time slices back desired. Each node has an associated conditional probability table which gives the probability of each of its values given the values of the nodes it is dependent on. An example of a simple DBN with nodes up to two time-slices back is shown in figure 2.3.



A-1	A=0	A=1
0	0.7	0.3
1	0.2	0.8

C-1	A	C=0	C=1
0	0	0.9	0.1
0	1	0.9	0.1
1	0	0.9	0.1
1	1	0.1	0.9

B-2	A-2	A-1	B=0	B=1
0	0	0	0.7	0.3
0	0	1	0.1	0.9
0	1	0	0.1	0.9
0	1	1	0.7	0.3
1	0	0	0.9	0.1
1	0	1	0.8	0.2
1	1	0	0.8	0.2
1	1	1	0.9	0.1

Figure 2.3: A simple DBN modeling the dependencies between the values of the variables A, B and C in the current time slice and the two previous time slices. Note only dependencies involving the variables in the current time slice are shown because all other dependencies would just be the time-shifted version of these. The CPTs shown assume that all the variables are binary indicator variables for simplicity.

A DBN's structure and conditional probability table values can be learned automatically from a database of sequences of the value of the variables [32]. One algorithm that does this is the REVEAL algorithm, which uses the mutual information between successive time slices to determine dependencies [33]. DBN structure learning has been used successfully in several domains, particularly in biology where it has been used to model gene expression data [34]. It has also been applied to learning simple behavior models of human driving [35].

I experimented with the Bayes Net Toolbox Matlab implementation of REVEAL [36], modifying it to include dependencies between more distant time slices and the current time slice. I also experimented with DBN learning program called LibB [37] and another by a company called Structured Data [38], but none of them scaled well to my application

out of the box.

In order to apply DBN learning to the problem of candidate event-sequence generation, I had to come up with a mapping between game-actions and DBN variables and values. There are multiple ways to do this but there are two extremes. The first is having each action or dialogue line map to its own binary indicator variable. These variables indicate whether or not each action or dialogue line appeared in a particular time slice. In other words, in each time slice, only one of the thousands of binary indicator variables would be 1 and the rest would be 0. The other extreme would be to have a single variable whose possible values correspond to all the dialogue lines and actions. I found the former case more intuitive because the dependency can be seen in the structure and the candidate event-sequences are more visible. In the latter case all the information is stored in one gigantic conditional probability table which would basically be a transition probability matrix. In this case the event-sequences would be less obvious and it would require development of an additional algorithm to construct event-sequences from the CPT.

The problem with the binary indicator variable case is that there are so many variables, but only one is going to be one at any given time. By default, DBN learning algorithms consider the complete CPT, meaning that a particular node's CPT would have an entry for every possible combination of values of it's parent nodes, even though in the binary indicator case only a small fraction of the combinations are actually possible (e.g. a combination with two parents in the same time-slice having a value of 1 is not possible). Furthermore, CPT estimation is a subroutine within a subroutine of typical DBN learning algorithms, it must be done for each node for each time a new candidate structure is being tested. Modifying the algorithm could allow it to take advantage of the constraint, but even still, the number of possible structures of a DBN grows factorially with the number of variables. Consequently, it would be difficult to scale up the binary indicator variable DBN method to the full set of game-actions, or even just the clusters of game-actions. Many learning algorithms do provide parameters to allow a trade-off between accuracy and speed. For example, you can set the maximum in-degree and out-degree of variable nodes, meaning the maximum number of node they can be dependent on or can depend on them respectively.

In practice, I found that with the Bayes Net Toolbox I had to limit the number of variables to the low double digits, rather than the 2000 or so I would have liked to use to represent the individual game-action clusters. In order to evaluate the effectiveness of DBNs to see if they were worth attempting to scale-up, I reduced the game logs to sequences of 19 action variables (shown in table 2.1), completely removing all dialogue. I experimented with using binary indicator variables for the actions as well as using the object of the action as the value of the variable. I found that the latter overemphasized variables with very few possible objects / values (e.g. the customer can only pay the bill). To correct for this, I artificially told the structure learning algorithms that all the variables has the same number of possible values. Having action variables whose values were the set of all object was the most effective. The results were promising, an

example structure is shown in figure 2.4, but ultimately the scalability issues and the challenge of mapping dialogue to variables led me to pursue sequential pattern mining as my means of candidate event-sequence generation.

Variable	Actor	Action	Variable	Actor	Action
1	Customer	Clean	11	Customer	Pay
2	Waitress	Clean	12	Customer	Pickup
3	Customer	Eat	13	Waitress	Pickup
4	Waitress	Eat	14	Customer	Put-down
5	Customer	Get-off	15	Waitress	Put-down
6	Waitress	Get-off	16	Customer	Sit-on
7	Customer	Give	17	Waitress	Sit-on
8	Waitress	Give	18	Customer	Touch
9	Customer	Look-at	19	Waitress	Touch
10	Waitress	Look-at			

Table 2.1: The reduced set of actions used to test DBN structure learning. All dialogue lines were removed and actions were combined based on just the actor and action involved. Very rare actions were also removed, such as the waitress paying.

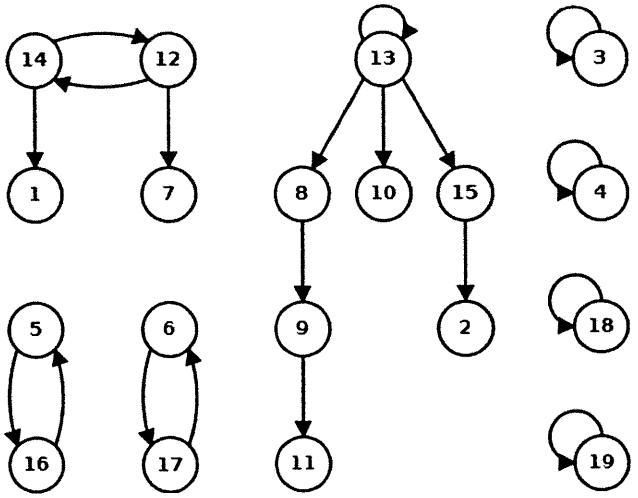


Figure 2.4: An example DBN learned using the REVEAL algorithm form the Bayes Net Toolbox with a Bayesian Information Criteria scoring function with a penalty of 1.0 (designed to limit over-fitting to the data) from 1000 games with a maximum in-degree of 3 from only the previous time slice. Only one set of variables is shown instead of one for each time slice because each dependency arrow indicates a dependency of one variable on the value of the other in the previous time slice. Notice the chain 13-8-9-11, which represents the waitress picking up something and giving it to the customer, then the customer looking at it and paying it. This chain represents the customer receiving and paying the bill events. If we were to examine the CPTs we would find other interesting chains, for example 13-8-9 is also the event of the customer getting the menu.

2.4 Clustering and Similarity Measurements

Another critical component of the event-discovery system is the several clustering stages. In addition to clustering event-sequence instances corresponding to the same event, I use clustering on words, dialogue lines and actions, to group them semantically and ultimately make event-sequence mining easier by reducing the number of unique

elements in the sequences. Similar to many of the systems described in sections 2.2.2 and 2.2.3, I first cluster words based primarily on the similarity of the contexts they appear in. For example, pie and tart occur in similar contexts such as "berry pie/tart" and "your pie/tart sir." Although the results of the word clustering are interesting by themselves, the purpose is to facilitate dialogue line clustering by allowing the surface similarity of dialogue lines to be measured in terms of semantic word clusters rather than individual words. This causes a quick reduction in the number of unique dialogue lines by combining those that are identical on the surface based on the sequence of word clusters they are composed of. This word-cluster-based surface similarity is combined with a context similarity measurement in order to cluster the dialogue lines themselves, while the actions are clustered in parallel. The motivation behind this is that the choice of sequential pattern mining algorithms as the means to generate candidate event-sequences only finds frequent event-sequences. Since about 80-90% of the dialogue lines in the logs are only used once (even when combining word-cluster-identical lines), this means that very few frequent sequences would contain any dialogue lines at all without further clustering.

In terms of the form of the input, there are two main categories of clustering algorithms. One type is algorithms that take as input a point in some feature space corresponding to each datum to be clustered. The other type is algorithms that take as input a similarity matrix representing how similar different pairs of data are. In all three of the event-sequence discovery cases (word clustering, dialogue and action clustering and sequence clustering), pair-wise similarity matrices are more intuitive. This is because I am combining several different similarity measures (e.g. surface similarity and context similarity) with different weights for each, so it is not obvious how this would map well to a feature space. Some of the popular clustering algorithms that work with similarity matrices are affinity propagation [39], the spectral clustering class of algorithms [40], and k-centers.

Based on the work of Grönqvist and Gunnarsson [41] and my own experiments with a wide variety of approaches, the main component of the similarity measurements for all three types of elements is based on the narrow contexts the elements appear in. Furthermore, these narrow contexts are in terms of the current cluster assignments of the elements rather than the elements themselves (see section 3.3.2 for more details of the implementation). This required that the similarity measurements be updated throughout the clustering process. Since spectral clustering methods do not use the similarity matrix directly, it would require frequent restarts of the clustering algorithm to use spectral clustering with cluster-based context similarity. Therefore affinity propagation was selected because it uses the similarities directly, and it is typically faster than k-centers because it is deterministic and does not require many random restarts.

My initial work focused on clustering words and dialogue lines simultaneously using a method I called *layered clustering*. As noted previously, word cluster assignments influenced the context similarity used to cluster words as well as the surface similarity used to cluster dialogue lines, while dialogue line cluster assignments influenced the context

similarity used in clustering dialogue lines. With just these links, word clustering and dialogue clustering can be done separately, with their similarities being updated during clustering. Layered clustering added a third type of similarity called parent similarity and a new link in which dialogue line cluster assignments determined the value of the parent similarity between words. This required words and dialogue lines to be clustered simultaneously, greatly increasing the complexity of the program as well as the time and space requirements.

The idea was that the context a word appears in within a game log itself is useful in determining the semantic meaning of the word, particularly in cases where the word-level context was very small (e.g. a single word utterance). The problem is some words appear in predominantly single word utterances (e.g. hello, yes, thanks), or other utterances in which the context of the utterance gives more information about the meaning of the word than the context within the utterance. This game-log context information could be given in terms of the context of the dialogue line the word appears in. The problem with that is that dialogue line context information is most useful in terms of dialogue line cluster assignments, and dialogue line cluster assignments are in-part determined by word cluster assignments. This created a catch-22 situation leading to the idea of clustering both simultaneously and having the cluster assignments of each iteration influence the similarities used in the next iteration. This is similar to the idea of expectation-maximization used to estimate parameter values in statistical models, where current parameter values and hidden variable values are used estiamte one-another iteratively [42].

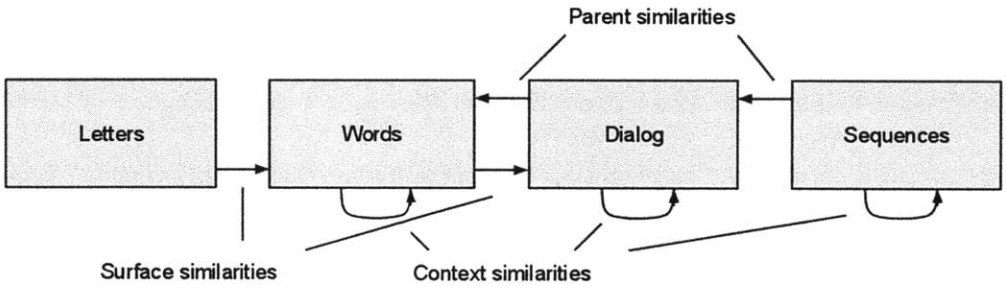


Figure 2.5: The interaction between neighboring layers in layered clustering in terms of the way current cluster assignments in one layer influence the similarity measurements in another layer.

The hope was also to ultimately add in a sequence layer, although this proved to be even more difficult due to the fact that the event-sequence discovery system requires logs to be rewritten in terms of dialogue and action clusters before sequences can be mined. Figure 2.5 shows how the different layers would influence one another. Initial experiments were promising, producing good results for word clusters and decent results for dialogue clusters when using just 10 games (examples are shown in figure 2.6). Ultimately however, the complexity and time and space requirements were deemed too high when dealing with the entire dataset to justify the limited benefit layered-clustering provided over separate clustering stages. The following subsections go into more detail about other research related to clustering words, dialogue lines and sequences separately.

tea	tooth	beer	order
marinara	staff	wine	stay
tart	quest	coffee	eat
sit	day	vodka	start
spagetti	typing		drink
speghetti	plate		pay
steak	cakes		hear
	meatballs		dc
	apologies		rape
	asdgas		
	messages		
	myself		

(a) Sample word clusters

C2W__and_this_bill	C2W_yes_please
W2C_and_the_berry_pie	W2C_yes_sir
C2W_and_the_pie	C2W_yes_i_am
C2W_and_the_lobster	C2W_yes
C2W_and_soup_de_jour	W2C_yes_faster
	C2W_tea_please
C2W_be_nice_to_the_chef	C2W_yes_thanks
C2W_ah_nice	
C2W_give_it_to_me	
C2W_bite_the_chef	

(b) Sample partial dialogue clusters

Figure 2.6: Sample output from layered clustering being run on data from 10 games. This shows some of the word clusters found and parts of some of the dialogue clusters found. C2W means the customer said the line to the waitress, and W2C means the waitress said the line to the customer.

2.4.1 Word Clustering

Word clustering is an important part of many natural language processing systems, it can be used to find synonyms for words [43], studying properties of language [41], as part of many dialogue systems [25] and more. Most word clustering, including that used in the event-sequence discovery system, seeks to group words by semantic similarity (how similar the meanings of the words are). Measuring word similarity in some form or mapping words to some feature space is a necessary prerequisite for word clustering and is also useful for many applications of its own, such as catching misspellings, learning word senses, automatically building or expanding thesauri etc. [6].

My initial efforts focused on similarity measurements that represent semantic similarity directly, such as thesauri-based methods including Lesk [44], the Wu & Palmer measurement [45], the Leacock and Chodorow method [46] and others [47]. These thesauri-based measurements approximate semantic similarity using pre-constructed semantic data structures such as WordNet. They measure the similarity of two words based on the relative positions of the words in the data structure as measured by, for example, the number of is-a relationships to reach a common root. These similarity measurements are limited in that they require a pre-built thesaurus which can be very difficult to create.

My experiments with these word similarity measures first applied spelling correction then part of speech tagging. From this I extracted lists of verbs and nouns which I applied a number of thesauri-based similarity measurement methods to prior to clustering. I encountered several problems, first the need for word sense disambiguation which I attempted to overcome by selecting the maximum similarity from all pairs of word senses to be the similarity for those two words. Another problem I encountered was that many of the words in my dataset were not present in WordNet or just did not work with some of the methods, a testament to the fact that the thesauri necessary for these methods are very difficult to create. Furthermore, such thesauri only contain certain types of words, like nouns and verbs, so

they could not be used to cluster all the words in my dataset. I also did some experimentation with measuring noun semantic similarity by comparing bag-of-word representations of the Wikipedia articles associated with the words based on the idea some semantic similarity measurements use of comparing the glosses of two words. I did not make much progress with this, but it might be interesting to pursue further in future work. I quickly discovered that context-based similarity measurements were much more effective for my particular case and later combined them with surface similarity measurements.

Context similarity and co-occurrence based clustering is a commonly used technique for word clustering in natural language processing applications. Chotimongkol and Rudnicky use context similarity-based (in the form of mutual information-based and Kullback-Liebler-based clustering) in their work with learning in and unsupervised dialogue system [25]. Grönqvist and Gunnarsson use very narrow context similarities based on the current assignment of words to clusters to hierarchically cluster Danish and Swedish words from spoken language corpora [41]. Nenadić, Spasić and Ananiadou used automatically discovered context patterns to find similarities between domain-specific terms [48]. Some of their other term clustering research has used context similarity as one component of the overall similarity measurement between words [49, 50]. I based my word clustering system off of this multitude of previous research and my own experimentation on my particular dataset. See section 3.3 for more details about my particular implementation.

2.4.2 Dialogue Clustering

Clustering of dialogue lines specifically has not been throughly explored in prior NLP research, but the similar problem of semantic sentence clustering has been tackled several times. Frey and Dueck used it to find exemplary sentences for their own research paper in order to demonstrate the effectiveness of their affinity propagation clustering method [39]. Radev et al. group sentences based on their information content as part of their multi-document summarization system MEAD [51]. Wang et al. used spectral clustering in conjunction with a sentence similarity matrix derived from semantic analysis based on WordNet-driven comparisons of the terms used in the sentences [52]. Zha used a similar spectral clustering method to group sentences by topic, but also incorporated the proximity of sentences into the clustering algorithm [53].

There are some differences between typical sentence clustering problems and the dialogue clustering step in event-sequence discovery. First of all, the nature of the dataset from which the sentences are taken are typically quite different. In The Restaurant Game the context surrounding a dialogue line can give a lot of information about its meaning. For example, "coming right up" and "I'll be back with that shortly" are two dialogue lines that mean the same thing, but have no obvious internal similarity that could be extracted automatically. However, the contexts they

appear in, typically right after a customer places an order and before getting the order from the chef, are very similar, demonstrating the semantic relationship between the two lines in the context of a restaurant.

Another difference is that the dialogue lines in The Restaurant Game can be short, such as "One please" or "Beer," and can't really be understood independently of the prior context, such as "Table for two?" or "What would you like to drink?" Although I did investigate some of the sentence clustering techniques used in other NLP research, many of them rely on the information contained within a sentence. The differences between the problems, coupled with my own early experiments led me to believe that a clustering system more similar to the one I decided to use for words would be more effective, see section 3.4 for more information on my specific implementation. In future work it would be interesting to investigate other methods for clustering the dialogue lines.

2.4.3 Event-Sequence Clustering

Sequence similarity measurements and clustering have been used in many sequence learning, prediction and categorization applications. For some examples see section 2.3. However, most of the sequence similarity problems have dealt with complete, contiguous sequences so the similarity measurements they have developed are not as applicable to the event-sequence clustering problem. For example, Lane and Brodley use edit-distance and alignment based sequence similarity measurements for their anomaly detection problem [10, 22]. Also Yang and Wang developed a generic sequence clustering algorithm that uses probabilistic suffix trees to group sequences [54].

The event-sequence clustering problem is different from most in the additional information available to use in computing similarities. There are additional properties of the sequence, such as the expansion factor and where it occurs in the game logs, as well as contextual information about where it appears in relation to other sequences that I take advantage of when clustering (see section 3.9 for more specifics). I also make use of surface similarity measurements based on edit distance similar to some of the other sequence clustering research, but it would be interesting to investigate some of the other surface similarity measurement options in the future.

2.5 Event-Sequence Labeling

Labeling is one of the final stages in the event-sequence discovery system. The purpose of the labeling step is two-fold, it provides data by which the system may be evaluated, and it allows for context and certain feature information to be gathered about the event-sequences in order to better cluster them. The labeling step itself consists of finding a valid (each action or dialogue line must be in at most one event-sequence) labeling of sequences in each log file.

Orkin developed the greedy RUN algorithm which, given a dictionary of all possible event-sequences, does a good job of labeling sequences in a new game during run-time [17]. The event-sequence discovery system runs completely off-line, allowing it to find the optimal labeling, rather than requiring it to resort to finding a good labeling.

There are two challenges associated with finding the optimal labeling. The first is determining the criteria that should be used to define optimality. Ideally the optimal labeling would be the one that most resembles the actual events that took place in the game. However, there is no way to score a labeling based on this without already having the logs labeled manually, which defeats the purpose of the entire system. Therefore other scoring systems must be developed based on this goal by thinking about the measurable properties the perfect labeling would have. First the labeling would likely cover most of the actions and dialogue lines in the log. Although people enjoy doing strange stuff sometimes, the majority of most logs is typical restaurant behavior which can be easily assigned to an event by a human annotator. Our system ideally should be able to do the same thing, although it is only working with frequent sequences, so it won't be able to catch as much as a human. This gives us one measurement we can use as part of our optimality score, the percentage of the log's elements covered by the labeling.

Other optimality scores can be derived from the intuition behind why and how humans are able to label logs easily. For example, the event-sequences in a labeling should be typical examples of those events. At the labeling step in the event-sequence discovery system, the sequences have not yet been clustered into event groups, but this does provide a basis for another component of the optimality score. We can measure the normality of the sequences in a labeling by looking at how similar their arrangement is to the average for that sequence in all the logs (see section 3.8 for the detailed implementation). Designing a system that iteratively labels logs and clusters sequences in order to be able to score normality better might be interesting work for the future.

The second challenge in selecting an optimal labeling is figuring out how to find it among all the valid labellings. There could potentially be many conflicting sequences, leading to a very large number of valid labellings for a single log, possibly making it difficult to enumerate them all. One possibility I explored was optimizing the labeling within sections of the log then combining them into an overall optimal labeling. The difficulty is that sequences can span potentially any part of the log so finding a way to split the log that does not break sequences into different sections may be impossible. One might still be able to make this work as an approximate method. Another option I experimented with was selecting one optimal sequence at a time for each set of conflicting sequences. The problem I found with this is that sets of conflicting sequences can be large, complicated and overlapping. For example, one sequence might conflict with three others none of which conflict with each other so you are actually choosing between one sequence and three sequences instead of selecting one out of four.

I decided that it was important to select the optimal labeling according to my scoring criteria, not just an estimated

optimal labeling. My reasoning was that there would be one fewer components that might impact the overall performance of the very complex event-sequence discovery system. Therefore, I ultimately decided to use the slow method of enumerating all possible valid labellings and finding the one that scored the highest (see section 3.8 for the implementation). Future work could look at improving this stage of the system.

2.6 Evaluation

The goal of the event-sequence discovery system is to find sequences corresponding to events and cluster them according to the event they correspond to. Determining how well the system is accomplishing this goal is a difficult task by itself. Of course one could observe the output and qualitatively evaluate it, and perhaps develop a quantitative evaluation system by having humans assign scores to the output in some way. However, it is surprisingly difficult to qualitatively evaluate the performance of the system, making a human-based scoring system (which would require many human subjects unfamiliar with the project) infeasible.

There are several reasons qualitative evaluation is difficult. The most straight-forward is that it is difficult to present the output in an easily interpretable way. Sequences are built up from action and dialogue clusters which are built up from word clusters, making it difficult to interpret exactly what is happening in a particular sequence. Of course on could list out all the specific instances of a sequence, but that brings us to the next difficulty, the amount of data. There are so many sequences that it is difficult to keep track of how many seem right and how many seem wrong. One possible solution to this would be to present only a small portion of the data to each human evaluator, but then it would be difficult to assess the clustering of the sequences, because clusterings can't really be evaluated well without looking at them in their entirety. These and other factors make it difficult for anyone besides those with expert knowledge of the system to qualitatively evaluate its performance, which makes a quantitative measure based on human evaluation difficult.

Without being able to have humans compare the output to their intuitive ideas of events, we are limited to comparing the output to something else. In conjunction with the development of EAT & RUN [17], Orkin manually labeled the event-sequences in 100 game logs. We also have Orkin's manual clustering of event-sequences in terms of the event label he applied to each. This allows us to evaluate the system by comparing its labeling of those 100 games logs to Orkin's manual labeling or by comparing its clusters to Orkin's. Orkin also worked to reduce unique dialogue lines by modeling dialogue as an unordered set of salient phrases. This creates a problem for comparing clusterings because the event-sequence discovery system deals with dialogue in a different way so there is no direct mapping between the model of sequences in the manual clustering with the model of sequences in the automatic clustering.

We chose to evaluate the system by comparing the automatic labeling with the manual (gold-standard) labeling in terms of sequence instances and event / cluster labels. One approach I explored was based on Allen relations [55]. These are the set of relationships that can exist between two temporal intervals in terms of the relative occurrence in time of their beginnings and ends. The idea was to compare the intervals defined by the event-sequences in the golden labellings with the intervals defined by the event-sequences in the automatic labellings based on these relations and compute a score depending on how much they aligned. The theory being that the more closely an automatic event aligns with a manual event, the better the labeling is, even if there are a few differences in the actions and dialogue lines contained in each.

I believe this is a very practical theory and would like to pursue this further in future work, but as of yet I have had challenges implementing such an evaluation system. The problem is that it is difficult to find a mapping between manual event-sequences and automatic event-sequences within a log, so it is hard to determine what to compare intervals between. The difficulty of the mapping stems from several factors, particularly the fact that sequences can overlap arbitrarily within a log and that the automatic labellings tend to be more sparse because they only label frequent sequences. Consequently, I chose to pursue other evaluation methods for the time being.

The evaluation method I settled on makes use of extrinsic clustering evaluation metrics to compare the golden labeling to the automatic labeling by treating the process of assigning game-actions to event-sequences as the process of assigning elements to clusters. The motivation stems from a relatively intuitive idea: if a pair of actions is in the same event-sequence in the golden labeling, then they ought to be in the same event-sequence in the automatic labeling, and vice-versa if they are not in the same event-sequence in the golden labeling, they should not be in the same event-sequence in the automatic labeling. By treating event-sequences as clusters, we can evaluate an automatic labeling by measuring how true these statements are overall in that labeling using extrinsic cluster evaluation metrics. For details on the implementation and metrics used see section 3.10.

Chapter 3

Implementation

This chapter details the design of the system developed in conjunction with this thesis. The first section describes the overall system, what the components are and how they fit together. The subsequent sections provide more detail on each of the components. The terms *stage* (because they are run in a particular order) and *component* (because they are all necessary parts of the overall system) are used interchangeably to refer to the separate parts of the system.

3.1 Overview

The system as a whole takes as input logs of human game play from the restaurant game (see figure 3.2) and produces several forms of output at different stages in the process, as detailed in figure 3.1. The outputs of the final stage are several scores that evaluate the overall performance of the system by comparing its labeling of logs to a gold-standard manual labeling on a particular subset of the logs. The system is composed of a number of stages, each one being its own executable or script written in C++ or Perl (although many share some of the same C++ code-base). The output of each stage is typically in the form of human-readable text files (or a human-readable version is output as well) and is fed as input into one or more future stages. Although, for the most-part, only the overall performance of the system was evaluated quantitatively, the intermediate output from the various components was evaluated qualitatively in order to modify algorithms and adjust their parameters. Furthermore, the outputs from many stages are useful and interesting in and of themselves, such as the word clusterings, or dialogue and action clusterings.

While figure 3.1 depicts the interaction of the components, their inputs, outputs and function are described briefly below. More detail on the algorithms and processes used in each stage are presented in the following sections.

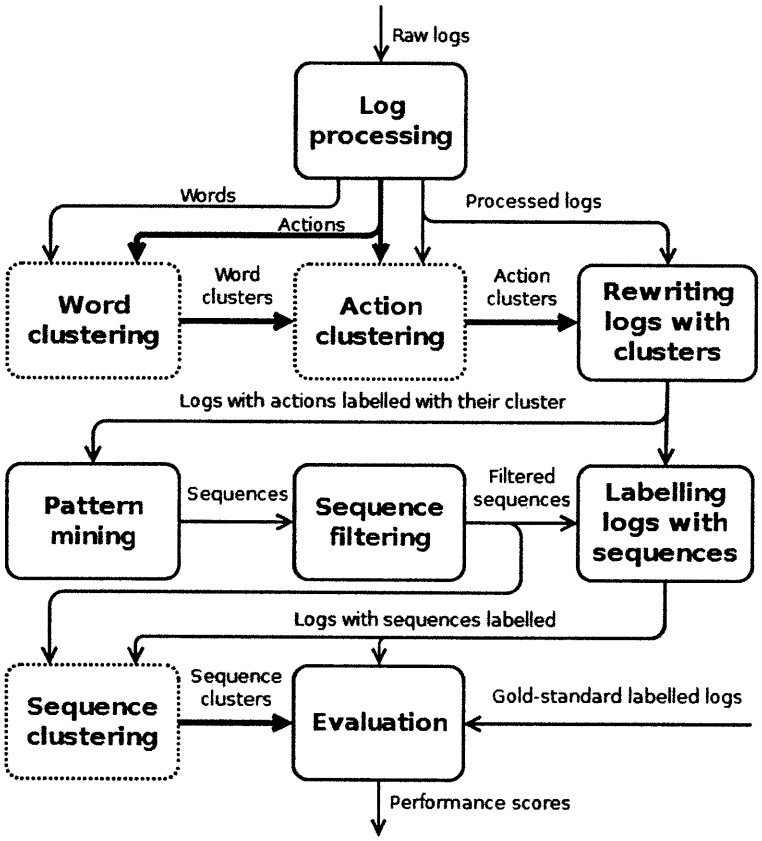


Figure 3.1: A flow chart showing the components and subsystems that make up the different stages of the event discovery system. For the components with dotted outlines, all the inputs to that component are passed along with the output to any components making use of the output. For example, the word clustering component produces clusters of words, but any other component that uses those clusters also receives the original words, because the two go hand-in-hand. Thick arrows represent multiple inputs/outputs.

1. **Log Processing**

   Input Raw logs

   Output Words, Actions, Processed logs

   The raw game logs (see figure 3.2a) are processed to extract sets of words and actions (including dialogue acts) appearing in the logs and to make them more human-readable and easier to process further (see figure 3.2c).

2. **Word Clustering**

   Input Words, Dialogue lines

   Output Word clusters

   Words are clustered based on surface similarity and context similarity.

3. **Action Clustering**

   Input Words / clusters, Dialogue lines, Actions, Processed logs

   Output Dialogue clusters, Action clusters

   Dialogue acts are clustered based on context similarity and surface similarity, while other actions are clustered based only on context.

4. **Rewriting Logs with Clusters**

   Input Processed logs, Dialogue lines / clusters, Actions / clusters

   Output Logs with actions labeled with their clusters

   Given processed logs and action cluster information append the cluster information to the logs so that sequence mining can run on clusters.

5. **Pattern Mining**

   Input Logs with actions labeled with their clusters

   Output Sequences

   A modified sequence mining algorithm finds frequent sequences that meet certain basic criteria.

6. **Sequence Filtering**

   Input Sequences

Output Filtered sequences

Sequences are further filtered to remove sequences unlikely to correspond to events.

7. **Labeling Logs with Sequences**

Input Logs with actions labeled with their clusters, Filtered sequences

Output Logs with sequences labeled

For each log, label a good, non-conflicting set of sequences in the log.

8. **Sequence Clustering**

Input Dialogue lines / clusters, Actions / clusters, Filtered sequences, Processed logs with sequences labeled

Output Sequence clusters

Cluster the sequences based on surface similarity and context similarity.

9. **Evaluation**

Input Filtered sequences / clusters, Logs with sequences labeled, Gold-standard labeled logs

Output Performance scores

Uses extrinsic cluster evaluation measures and other measure to produce a set of scores comparing the system-driven labeling of sequences to the gold-standard labeling of sequences in a subset of the logs.

3.2 Log Processing

This is the first stage, which converts the raw log files into a more usable form. This stage actually consists of multiple steps implemented in several different Java and C++ programs. The first few steps are taken from earlier work done by Orkin. These steps involve filtering out all the unused information from the logs (e.g. debug information, movement information etc.) leaving only dialogue acts and actions. Additionally, an action lexicon is created which associates an index to the actor, object, state changes and verb of a particular action. The lexicon is reduced by limited clustering of game objects based on the similarity of the actions they are used in (e.g. all the different chair objects, all the different forms of the wine object etc.). The logs are then rewritten in a reduced form with actions replaced by indices into the action lexicon. These initial steps, borrowed from Orkin's earlier work take the logs from their raw form as in figure 3.2a to an intermediate form as in figure 3.2b.

```
...
[COGLOG] 15097868 EVENT WAITRESS(Player)
         FACING CHEF(DBChef)
[COGLOG] 15097868 EVENT WAITRESS(Player)
         FACING BARTENDER(DBBartender)
[COGLOG] 15097868 EVENT WAITRESS(Player)
         FACING CUSTOMER(Player)
[COGLOG] 15097868 STATECHANGE WAITRESS(Player)
FORWARDDIR="(-0.852038 0.254967 -0.457191)"
[COGLOG] 15097868 STATECHANGE WAITRESS(Player)
ATPOS="(311.714 367.582 240.556)"
[COGLOG] 15097868 SPEECHACT WAITRESS(Player)
         "Here is your menu."
[WAITRESS] Here is your menu.
SELECTED: WAITRESS CUSTOMER 1208 310.513 367.943
243.188 0.684043 0.545956 0.483754
Mapping string: onGive to index: 16
REACH: WAITRESS CUSTOMER 2
[COGLOG] 15101212 STATECHANGE WAITRESS(Player)
ATPOS="(311.714 367.582 240.556)"
[COGLOG] 15101212 ACTION WAITRESS(Player)
         GIVE dynamic1216(DBMenu)
[COGLOG] 15101212 STATECHANGE dynamic1216(DBMenu)
ATTACHEDTO="CUSTOMER(Player)"
...
```

(a) Raw log

```
...
ASSERTION_GIVE_MENU     <TAG=TIME 15097868>
         // 15097868 WAITRESS: "Here is your menu."
92.1.1   <TAG=TIME 15101212>
36.1.2   <TAG=TIME 15112477>
DIRECTIVE_PREPARE_FOOD  <TAG=TIME 15129852>
         // 15129852 WAITRESS: "Water, please."
52.4.1   <TAG=TIME 15130852>
94.4.1   <TAG=TIME 15133024>
EXPRESSIVE_THANKS_OTHER <TAG=TIME 15140868>
         // 15140868 CUSTOMER: "thank you,"
...
```

(b) Intermediate log

```
...
10000.500142     <TAG=TIME 15097868>
92.1.1   <TAG=TIME 15101212>
36.1.2   <TAG=TIME 15112477>
10000.500143     <TAG=TIME 15129852>
52.4.1   <TAG=TIME 15130852>
94.4.1   <TAG=TIME 15133024>
10001.10002      <TAG=TIME 15140868>
...
```

(c) Processed log

Figure 3.2: Example excerpts of the raw log input, intermediate form (after using Orkin's tools), and final processed log output of stage one. While the later two excerpts correspond to the same set of actions and dialogue acts, the first excerpt corresponds only to the first two lines of the other two.

A similar process is repeated on the logs, this time focusing on dialogue acts and words. The intermediate form logs are reread and two things are generated simultaneously, a dictionary of words and a list of all unique dialogue lines. The word dictionary contains: words; their occurrence count; and their Inverse Document Frequency (IDF), treating dialogue lines as documents. The dialogue line files contain: the dialogue lines, in the form of a list of word dictionary indices; whether it was said by the customer or waitress; and the total number of occurrences. While those data files are being generated the logs are being rewritten to replace the dialogue lines with indices of unique dialogue lines. This takes us to the final processed log shown in figure 3.2c.

Separating the dialogue lines and actions from the log files makes it easier to deal with each. Having logs be a sequence of integer identifiers makes pattern mining easier, and it allows us to treat all instances of a unique dialogue line across different logs as references to the same object. Furthermore, words can be clustered using just the word file and the dialogue file as described in the following section.

3.3 Word Clustering

The goal of this stage is to group words based on their semantics. This is a useful first step for grouping together dialogue lines based on their semantic meaning. The following sub sections describe the algorithms used and the motivation for selecting those particular algorithms.

3.3.1 Clustering Algorithm

The word clustering stage is run second, it uses affinity propagation [39] to form hard clusters of words. Affinity propagation is a method of clustering that takes in a possibly incomplete matrix of similarities between data points and outputs an exemplar for each data point. The exemplar is the data point that best represents the cluster of points that have selected it. The algorithm for selecting an exemplar for each data point consists of iterative message passing. One message, responsibility, represents how much one data point wants another to be its exemplar out of other possible exemplar. The other message, availability, represents how much one data point wants to be an exemplar for another data point based on support from other points selecting itself as an exemplar. The messages are computed based on messages received from other data point in the previous iteration as well as the pair-wise similarities of the data points.

Affinity propagation was chosen as the clustering algorithm for all clustering steps in the system because of several desirable properties it has.

It uses pair-wise similarities as input

This is useful because when dealing with words, dialogue lines, sequences etc. The feature space is not obvious, making computing pair-wise similarities easier to work with and more intuitive.

It is deterministic

Unlike k-means and some other algorithms with random components, affinity propagation is deterministic. This is useful because a single run of many clustering stages can take a long time and non-deterministic algorithms often require multiple or many runs to perform best.

Similarities can be updated while running

Unlike spectral clustering, which reduces the dimensionality of the similarity matrix rather than using it directly, affinity propagation uses the original similarity matrix when computing messages at each iteration. This allows similarities to be modified during clustering based on the current soft assignments of elements to clusters.

Affinity propagation is also quite efficient for our purposes, the duration of clustering being dominated by the simi-

larity updates. Although the final output of the clustering algorithm is a set of hard assignments to exemplars, soft-assignments are used while the clustering is still running to update the similarities.

3.3.2 Similarity Measurements

Affinity propagation takes as input a matrix of similarities, which are updated during clustering in this system. The similarity between words is based on two factors, a surface similarity and a context similarity, each one measure on a scale from 0 to 1. The final similarity, also on a scale form 0 to 1, combines these two scores based on an algorithm parameter determining the weight of each. Surface similarity is a measure of how alike two words appear. It is intended to ensure typos and misspellings are clustered together. Context similarity is a measure of how similarly two words are used in terms of the words that appear next to them. Context similarity is used to represent semantic similarity. Other methods of measuring semantic similarity require outside data sources, and don't work on all word types, as described in section 2.4.1. Only the context similarity is updated during clustering by using the current clusters of words, rather than the words themselves, when evaluating contexts.

Surface Similarity

The surface similarity is derived from Levenshtein distance [56], also sometimes called edit distance. Levenshtein distance is the number of character insertions, swaps or deletes it takes to get from one string to another. A fairly straight-forward dynamic programming algorithm can be used to compute levenshtein distance [57]. An example of computing levenshtein distance based on this algorithm is shown for the words "salmon" and "slaomon" in figure 3.3.

However, levenshtein distance increases with dissimilarity. Therefore, to create a surface similarity measurement I normalize it so be on a scale from 0 to 1 and subtract it from 1. Normalization is done by dividing the Levenshtein distance by the maximum possible for the two words. The maximum possible edit distance occurs when every character of the shorter word must be swapped and the remaining characters of the longer word inserted. This corresponds to the length of the longer word.

Context Similarity

The context similarity measurement I use is derived from the context similarity section of the Jurafsky and Martin NLP textbook [6]. The basic idea is that each word has a context vector representing all the contexts it appears in. Thus the context similarity of two words can be measured by comparing their context vectors. They pointed out that there are several options for both constructing the vectors and comparing the vectors. Each dimension of a context

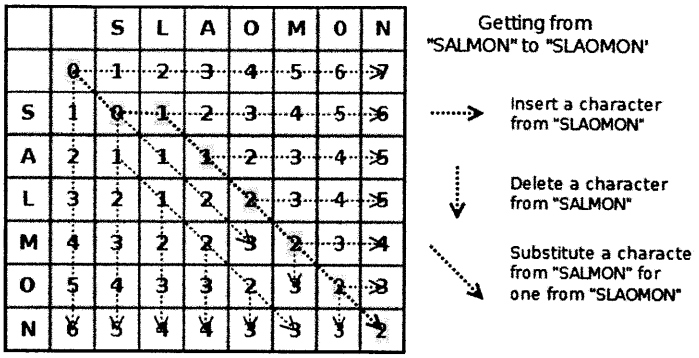


Figure 3.3: This figure shows the dynamic programming process the system uses to compute the levenshtein distance between the words "salmon" and "slaomon." Each entry D(i,j) in the grid gives the number of operations needed to get from the first i characters of "salmon" to the first j characters of "slaomon". Thus the complete edit distance between the words is 2 as seen in the bottom right of the table. If we think of the grid as showing how to get from "salmon" to "slaomon," a move to the right represents inserting a character from "slaomon;" a move downward represents deleting a character from "salmon;" and a diagonal move downward and to the right represents substituting a character from "salmon" with one from "slaomon." Each move has a cost of 1, with the exception of a substitution when the two characters are the same. The algorithm then is basically just finding the least-cost path from the starting location in the top-left to the ending location in the bottom-right. The least-cost paths to each grid cell are shown in the figure by the dotted lines. The green grid cells show the least cost path of the solution. This shows that it only takes two edits to get from "salmon" to "slaomon." The first, an insertion of an L between S and A and the second a substitution of an O for an L. There are often multiple least-cost paths, but only one is marked for each cell in the figure.

vector represents a particular context feature. For example: word A appears immediately prior; word B appears two words before; word B appears two words before and word A appears immediately prior etc. Furthermore, each value in a particular word's context vector represents the affinity between that word and that context feature. Therefore method of context similarity measurement has three components: context feature choice; affinity function; and vector comparison function.

Curran has shown that t-test affinity as an affinity function and Dice similarity as a vector comparison function are good choices when trying to use context similarity to relate words semantically [43]. The t-test affinity function is:

$$\text{assoc}_{\text{t-test}}(w, f) = \frac{\text{P}(w, f) - \text{P}(w)\text{P}(f)}{\sqrt{\text{P}(w)\text{P}(f)}} \tag{3.1}$$

where f is the context feature and w is the word. P(w,f) is the probability that word w occurs with context feature f (i.e. the number of times they appear together divided by the total number of times and feature and word appear together). P(w) is the probability of word w (i.e. the frequency of the word). P(f) is the probability of feature f (i.e. the frequency of the feature). The Dice vector similarity function is:

$$\text{sim}_{\text{Dice}}(\vec{v}, \vec{w}) = \frac{2 \times \sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} v_i + w_i} \tag{3.2}$$

where $\vec{v}$ and $\vec{w}$ are the context vectors of the two words being compared and v_i and w_i are the corresponding components of the vectors (i.e. the t-test affinity between each word and the same context feature). Dice is not only effective, but efficient to compute as compared to dot product for example.

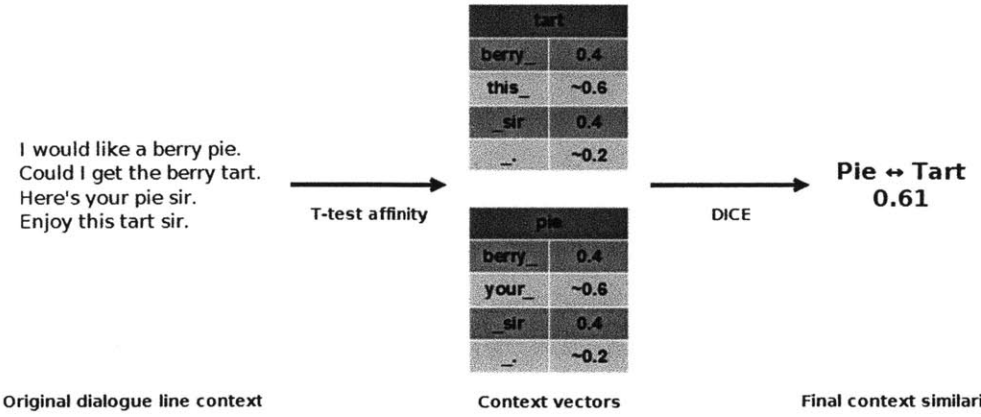


Figure 3.4: This shows how the context similarity of "pie" and "tart" is computed from the sample original contexts they appear in on the left. First a context vector is generated for each word which stores the t-test affinity between that word an each context feature it appeared with. Then the context vectors of the two words are compared using Dice to produce the final context similarity.

In terms of the selection of context features, [41] suggests that very small contexts can be sufficient when clustering words and interpreting context in terms of the clusters. Building on this idea, the context features in this system are re-computed after every few iterations of the clustering algorithm, and only concern one or two words. For example, some context features might be: a word from cluster A appears immediately prior; a word from cluster B appears two words before and a word from cluster A appears one word before. The weight associated with a context feature is the frequency it occurs with, which is used in the calculation of t-test affinity (3.1). The process of updating the context similarity is as follows:

1. Get the current soft-assignment of words to clusters (these sum to 1 over all the clusters a word is in).

    - The clustering algorithm produces a list of exemplars, and rather than assigning each word to the most similar exemplar as is done for hard assignments, the assignment is distributed among the N most similar exemplars, weighted by the similarity to the exemplar.

2. Create a new set of context features based on word clusters.

    - When reading in the dialogue lines, the context feature prototypes are stored in terms of the exact words. When generating the actual context features, the words' clusters are referenced and the weight associated with the prototype is distributed among the appropriate context features. So one context prototype will contribute weight to multiple context features, and one context feature will receive weight from many

context prototypes. There are much fewer features than prototypes because there are fewer clusters of words than words.

3. Create a new context vector for each word using the new context features.

   - Words point to the context prototypes that they are associated with. In order to generate context vectors based on the context features the words distribute their affinity to a prototype across the features that it corresponds to.

4. Compute the context similarity for all pairs of words that we are updating.

   - To make the process more efficient, as clustering progresses, similarity stops being computed for pairs of words that have remained relatively dissimilar.

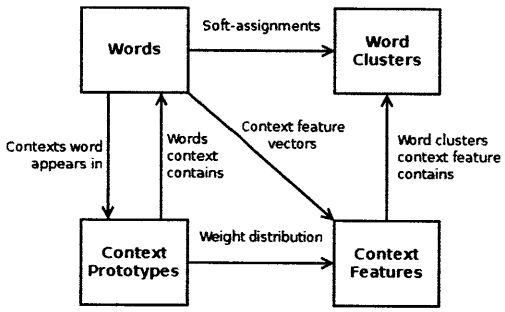


Figure 3.5: This shows how the various objects used to compute word context similarity are linked together. A context prototype such as "berry _" in figure 3.4, has references to the words it contains, in this case "berry". After computing new context features based on current cluster assignments, the prototype also has pointers to the context features it has been distributed among. Context features only refer to word clusters. For example, "berry" might have a soft-assignment of 0.9 to the "fruit" cluster and 0.1 to the "flavor" cluster, so the context prototype for "berry _" would be distributed among the context features "fruit _" and "flavor _" with a weight of 0.9 and 0.1 respectively. Word objects, such as "tart" refer to the context prototypes they are associated with (e.g. "berry _") in order to update their context feature vectors based on how the prototype is distributed among the features.

Figure 3.5 shows how words, word clusters, context prototypes and context features are related. While figure 3.4 shows a simple example of computing context similarity using word-based context features (rather than word-cluster based as is actually used).

3.4 Action Clustering

The action clustering stage produces three sets of clusters, waitress dialogue line clusters, customer dialogue line clusters and action clusters. The process is pretty much identical to the one used in the word clustering stage (3.3) with a few small differences.

First, because the output is separate sets of clusters for each type of object, the clustering of the objects is done in separate instances of the clustering algorithm, not all together. However, because all the objects use the same set of context features, the clustering of the objects must be stopped frequently in order to update the context features across the board. Context features are still very limited in size, and are obtained from the processed logs.

In this stage, surface similarity is only computed for dialogue lines and is still only a minor factor in the overall similarity computed. Surface similarity is not computed for actions for two reasons: actions are much more limited in number and form; and some surface similarity-based combination of actions has already taken place prior to this stage during log processing. There are some changes to the way surface similarity is computed. The biggest change is that the base unit of levenshtein distance is now a word rather than a character. This allows for more complicated weighting of the cost of insertions, substitutions and deletes. Instead of being only either 0 or 1, the cost of an edit is now based on the TF-IDF weight of the words involved in the edit. The last significant difference is that a substitution has a cost of 0 if the two words are in the same cluster, not just the same word.

3.5 Rewriting Logs with Clusters

This stage is very simple. Given the clusters output by the previous stage, it prepends a cluster id to every action and dialogue line in every processed log. No information is removed, the cluster an action or dialogue line belongs to is just added to the front. This allows the pattern mining performed in the next stage to work with clusters rather than individual unique action and dialogue lines. This is necessary since the pattern mining only finds frequent subsequences, and the vast majority of dialogue lines occur only once or twice.

3.6 Pattern Mining

As discussed in the background section 2.3, a lot of experimentation and research went into selecting and modifying the pattern mining algorithm, arguably one of the most important components of the system. Ultimately, the implementation used is a modified version of the PLWAP algorithm presented in [29]. PLWAP is used in the system for two reasons: it is relatively efficient in both time and space; and the data structure used to assist with mining allows for a lot of early pruning of sequences unlikely to correspond to events. This stage was built off the open source PLWAP implementation released by the original creators of the algorithm [58]. Most of the modifications to the algorithm deal with pruning non-event-sequences early. By default, PLWAP counts support for patterns in terms of how many sequences (in this case log files) they appear in. Since many events can be repeated multiple times in the same log

(such as eating, ordering cleaning dishes etc.), the algorithm was modified to compute support count in terms of the total number of independent (not intersecting) times a pattern appears in the sequence database. The other modifications, designed to help with pruning are described below. An example of the modified algorithm running on a simple sequence database is shown in figure 3.7.

3.6.1 Pattern Pruning

The original PLWAP algorithm already is designed to select only patterns that occur a certain minimum number of times and are less than a certain maximum length. The minimum frequency criteria is useful to the system because the goal of this stage is to find the most typical event-sequences. The maximum length criteria is useful because valid event-sequences usually consist of only a few game-actions. It turns out that with some slight modification, PLWAP can also prune patterns during mining based on other criteria useful to the system. This greatly increases the speed of pattern mining and also reduces the number of candidate event-sequences that have to go through the additional filtering stage. The sub-sections below describe two other forms of pruning which take place during pattern mining.

Besides the minimum frequency parameter which was chosen through experimentation, the parameters which control the various pruning mechanisms were chosen based on examination of the values for real event-sequences from the manually-labeled golden log set. The values used for each parameter, including those used to do post-mining filtering, are given in appendix B in section B.3.

Expansion Factor

One of the reasons that finding event-sequences in a log is hard is that the action and dialogue lines that make up the sequences might not be consecutive. Action and dialogue lines corresponding to other events or no event at all may be interleaved with those of a single event. However, it is still the case that an event-sequence is complete, no other actions or dialogue lines are needed in order to complete that particular event. Furthermore, this system works only with the lowest level of events in the event hierarchy, which are predominantly simple, short-term behaviors such as ordering food or cleaning a plate. As a result, one would expect that the majority of the components of a event-sequence tend to be consecutive or, at least close together. Or in other words, the number of game-actions between the first and last game-action of a event-sequence (the *real length* of the event in the log) should not be significantly more than the total number of game-actions in the event (the *sequence length* of the event). I refer to the ratio of the real length of a particular instance of an event-sequence to the sequence length as the expansion factor of that instance (see figure 3.6). As expected, the average expansion factor found for real event-sequences in the golden logs was relatively

low, mostly varying between 1.0 and 1.6 depending on the event label [1]. Certainly some event instances are outliers with expansion factors as high as 10, but since we are only mining frequent patterns and using the average expansion factor, they do not have a big impact on the pruning.

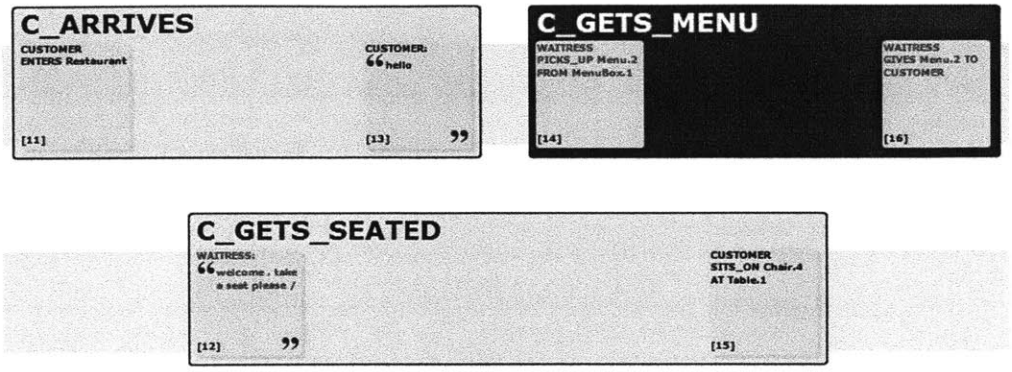


Figure 3.6: This excerpt from a manually-labeled golden log shows three event-sequences instances corresponding to the customer arriving, the customer getting a menu and the customer being seated. The blue and green instances have expansion factors of 1.5 while the pink instance has an expansion factor of 2.0.

The pruning of sequences based on expansion factor that occurs during mining is only approximate. This is because, in order to make the mining more efficient, sequences are pruned as soon as a prefix of the sequence has an excessive average expansion factor. The ideal way to prune based on expansion factor would be to only prune completed sequences, because the completed sequence will likely have a different expansion factor from its prefixes. However, if this was the only expansion factor filtering that took place, it would not speed up pattern mining at all and only be necessary in the post-mining filtering stage. Therefore, the system performs expansion factor filtering twice, once during mining with a higher maximum expansion factor, and once in the filtering stage with the maximum expansion factor chosen to reflect the observed values from the golden logs. In practice though, pruning just during mining with the lower maximum does not seem to have much of an impact on the final set of filtered sequences. This is logical because just as one would not expect a valid event-sequence to have a high average expansion factor, one would not expect and subsequence of that event-sequence to have a high average expansion factor for the same reasons.

In order to properly compute the average real length of a pattern during mining, the algorithm had to be slightly modified to select the shortest instance of a pattern when more than one intersected in the same log. This was handled by essentially throwing away the prefix that started earlier when two prefixes were extended to the same node in the mining trie. The real length of a pattern instance was easy to compute from the binary encodings of the first and last element of the sequence. The real length is just the difference in the number of set bits in the bit-codes of the two nodes, because each additional 1 appended to the code represents another level deeper in the trie.

[1] The "Customer Departs" event had a much higher average expansion factor than all the other events (greater than 5.0). This calls into question, how suitable it is to be a lowest level event in its current form.

Transition Probability

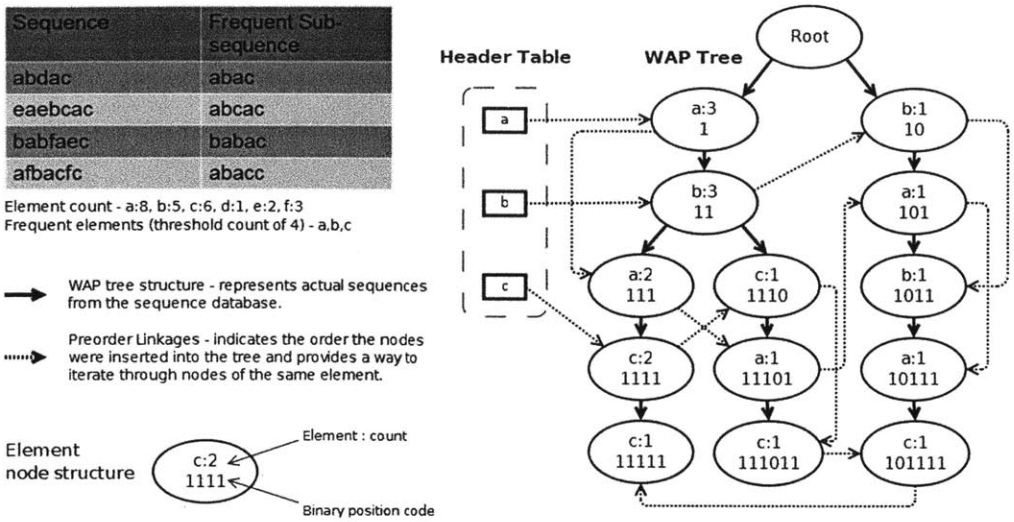
Based on the same argument used above, that valid event-sequences of game-actions are complete and don't need any other game-actions inserted, patterns are also pruned using the transition probability between game-actions. Patterns that contain consecutive game-actions with transition probabilities of near zero are removed during mining as soon as the low transition probability is encountered. This is because such a low transition probability implies that even though this pair of game-actions is frequent, there is always, or almost always, another game-action in between the two, suggesting that the corresponding event-sequence is incomplete. This method of pruning is primarily used to filter out sequences in which some of the actions can not occur consecutively due to restrictions of the game engine. For example, in the restaurant game it is impossible to have a pick-up action twice in a row because the first item to be picked-up must be given away or put down before a new item can be picked-up.

The transition probability of all pairs of game-actions was computed and stored in a matrix indexed by the ids of the game-actions at the same time that the trie for mining was constructed. Then they were looked up for comparison to the minimum threshold during the actual mining

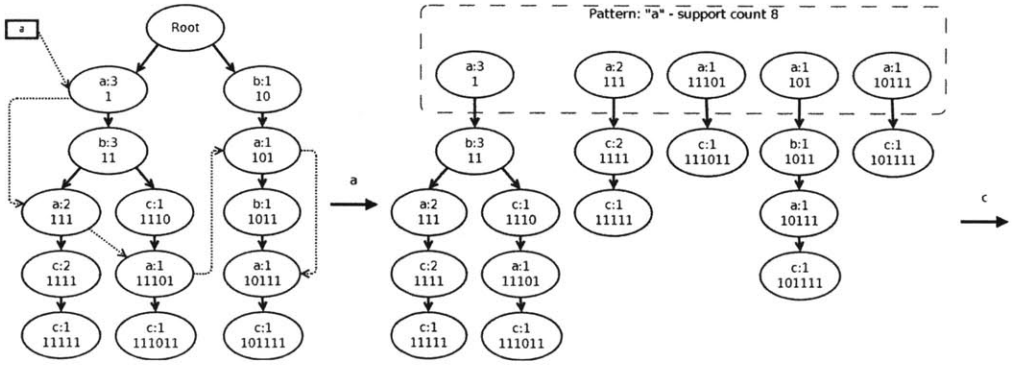
3.7 Additional Sequence Filtering

The filtering that takes place during pattern mining does not filter out many of the non-event-sequences. Some of the filtering is easiest to do after mining is complete, which is where this stage comes in. Section 3.6.1 refers to an additional pass of expansion-factor based filtering that takes place at this stage. This is one of several minor filters that catch a small number of bad sequences. The most important filters at this stage use the notion of closed-sequences.

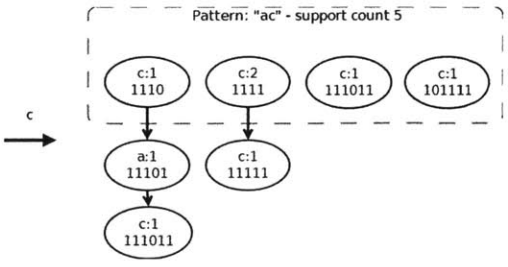
In sequential pattern mining, a closed sequence is a sequence that is not the sub-sequence of any sequence with the same count. This notion is useful because by this definition, a non-closed sequence is one that always occurs as part of a larger sequence. For example, suppose you have a sequences database in which the subsequence ABA only occurs within instances of the subsequence ABAC. In this case, ABA is not a closed sequence because it has the same count as a super sequence in the database, ABAC. It is easy to see how capturing only closed sequences can help in this system, eliminating incomplete sub-sequences of event-sequences which would otherwise pass all the other filtering criteria. However, due to the great variety in behavior and dialogue, I have found that using a slightly broader notion of closed sequences, which I refer to as pseudo-closed sequences, to be more useful. Instead of requiring the counts to be identical between a super and sub-sequence, I require that they be within the minimum frequency count of one another. That is, if I am mining only sequences that occur at least 50 times, I remove all sequences with a support count that is less than 50 away from the support count of one of their super-sequences.



(a) First the sequence database is rewritten, removing all infrequent elements. Then the frequent sub-sequence database is read into a trie where nodes of the same element are linked together in the order in which they were added to the trie. A special header table is created to reference the beginning of each of these preorder-linked lists. Each node in the trie has a count, the number of sequences in the database that pass through that node. Each node also has a binary position code which allows for easy comparison between nodes to see where they are in relation to one another in the trie during mining.



(b) An element is selected from the header list and the preorder linkages are used to find all the subtries that begin with that element. Note that these are all the sub tries beginning with the element, even if one is a subtrie of another. This is not how the normal PLWAP algorithm works and is part of the modification to count patterns based on their total number of occurrences. The count contained in each of the new roots is summed up and compare to the minimum frequency threshold to determine if this is a current sequence and if mining should continue.



Another element is selected from the header table, but this time we only select subtries from the previous set of subtries. Furthermore, each new subtrie is only included once, even if the previous set of subtries included it multiple times. For the purposes of real-length calculation, the new subtrie is associated with the smallest of the old subtries that contains it. For example, in the case shown the second new subtrie (c-c), is contained in the both of the first two subtries of the previous set. In this new set it is only included once and associated with the subtrie (a-c-c).

(c)

Figure 3.7: This is an example of how the modified PLWAP algorithm used in this system works (without the various pruning steps besides frequency). It is running on a simple sequence database mining the frequent sequences "a" and "ac", assuming a minimum frequency of 4. This example is borrowed from [58].

There are sequential pattern mining algorithms that only mine closed sequences and could be adapted to mine pseudo-closed sequences (i.e. using my looser definition of closed), but PLWAP has many nice properties as described in section 3.6 and I am using my notion of pseudo-closed sequences for other forms of filtering at the same time. In addition to removing all non-pseudo-closed sequences, I also remove sequences that are combinations of other pseudo-closed sequences. This removes sequences corresponding to two different event-sequences that occur back-to-back (e.g. we might mine a sequence corresponding to the customer entering and then getting seated because these often occur next to each other).

The filtering in place is not perfect, many event-sequences are filtered out (in particular infrequent ones), and many non-event-sequences slip by. There is a lot of room for improvement and I have a lot of ideas for what should be pursued in future work (see section 5.2.4), but this prototype system performs fairly well over-all as discussed in section 4.5.

3.8 Labeling Logs with Sequences

Once we have a list of likely event-sequences, we can label the sequences that appear in the game logs. This serves two purposes: to provide contextual and other information about the sequences for use in clustering; to allow for comparison to the golden manually-labeled logs in order to evaluate the systems overall performance. Those two purposes are the final two stages of the system and will be described in the next two sections. This section describes the process of labeling the sequences in the logs.

The problem with labeling the mined sequences that occur in a particular log is that there may be many different ways to do it. Many sequences share one or more elements with one another. Therefore, several sequences may occur in the same log that use the same element instance. However, in a valid labeling of event-sequences in a log, each element can occur in at most one event-sequence. Therefore, a single log may have multiple valid event-sequences labellings given a particular event-sequence list. The challenge is to select the best labeling for a particular log.

Finding all the valid labellings for a log is time-intensive but straight-forward:

1. Load all the event-sequences into a trie.
2. Iterate through all sequences in the log. Use the trie to determine if they are valid event-sequences.
3. Find the set of actions and dialogue lines that are part of more than one event-sequence.
4. Find all the valid sequence labellings, by constructing a set of sequences to include for each. To find a single

valid labeling the process is:

(a) Start at the first action or dialogue line in multiple event-sequences. Assign it one of its sequences and place the others in a set of sequences to exclude.

(b) Continue to the next element in multiple sequences. If the element has any sequence not in the set to exclude, assign it one of those sequences and place the rest in the exclusion set.

(c) Repeat 2 until all multiple sequence elements have been assigned a single sequence or have no sequences not in the exclusion set.

(d) Iterate through all the dialogue lines and actions, creating a labeling by assigning each element to its not-excluded sequence if it has one.

To find all the valid labellings, sub-steps 1 and 2 iterate through all sequences available for choosing at each element.

The tricky bit is determining what makes a labeling good and finding a way to quantitatively measure it in order to easily determine the best labeling. I measure the goodness of a labeling based on the following:

Coverage

The percentage of actions and dialogue lines in the log that are part of an event-sequence in the labeling. In general, the majority of the actions in a log can be attributed to some event-sequence when labeled manually. We want our automatic labeling to reflect this so we consider labellings with greater coverage to be better.

Normality of event instances

How normal the particular instances of event-sequences are relative to the average of all the instances of that each event-sequence (e.g. does this instance have a much higher expansion factor). This is computed as a score from 0 to 1 for each instance and then averaged to get a score for the entire labeling. Each instance's normality is simply the normalized difference in expansion factor between the instance and the average. Other factors of normality (e.g. context and log position) would require prior log labeling or bootstrapping.

Each labelling's coverage and normality of event-sequence instances is computed on a scale from 0 to 1 and then combined into an overall goodness score for the labeling. Coverage is given the greater weight in the final goodness score. There are other notions of goodness that could be factored in. A good one might be some holistic measure that looks at the typicality of the totality of the labeling rather than individual event instances. For example, how normal the overall order and number of events is. However, this would require much more computation and complicated

algorithms to bootstrap what is typical, or an external model of typicality based on manual labellings or something similar.

One could also imagine measuring goodness in terms of the overall system evaluation score produced by a particular labeling. This idea also has several drawbacks. First, it would only work on logs that have been manually labeled and could be compared against. Second, the coupling of the labeling and the evaluation would prevent examination of the effects of changing either independently on the final results. Third, there would be a bit of a boot-strapping problem because part of evaluation requires sequences clusters, and sequences clustering requires labeled logs, etc. The main benefit of such an approach would be to allow for evaluation of just the sequence mining and clustering (assuming the boot-strapping problem is solved properly) portion of the system, without having to worry about the possible effects the choice of labeling is having on evaluation. However, since I am also interested in how well the labeling stage works, I chose not to pursue this as of yet, though it would be interesting to look at in the future.

3.9 Sequence Clustering

This clustering stage is very similar to the previous two clustering stages. The input is the mined and filtered set of possible event-sequences, and the logs labeled with event-sequences. The output is a clustering of the event-sequences in conjunction with an exemplar for each cluster. Levenshtein difference is computed pretty much the same way as in section 3.4. In this case the units are action and dialogue clusters and the TF-IDF factor for each is computed as the sequence database is read in.

One major difference between sequence clustering and the other clusterings is the context features used for context similarity (see section 3.3.2). Instead of the features being just which other sequences occur immediately before or after the particular sequence, the context relationships are more complex. So now instead of just two possible feature relationships, immediately before or immediately after, there are eight. For a given pair of sequences A and B that occur near each other the possible context features that A could have along with an example of each is shown in table 3.1.

These context relationships are based on Allen relations [55]. There are not as many as there are Allen relations because of the discrete nature of the log. Within a single labeling of a log, a single action can not be part of multiple event-sequences. This eliminates the Allen relations of starts, finishes and equals. One other difference is that for my definitions of before and after I limit it to being at most a few actions apart. This helps keep the size of context feature vectors down and also helps maintain the notion of these representing contexts. However, it might be useful to include more relationships for different lengths of time before and after in future

Context Relationship	Allen Relation	Example	Description
B is nearby and precedes	B before A	B_BB__AA__A	The last action of B is a few actions before the first action of A.
B precedes and is adjacent to	B meets A	B_BBA_A_A	The last action of B is immediately before the first action of A.
B overlaps and precedes	B overlaps A	B_BB_ABA_B__A	B starts before A but ends during A.
B covers	A during B	B_BB__AA_A_BB	B starts before A and ends after A.
B is covered by	B during A	A_B_BA_B_AA__A	B starts after A and ends before B.
B overlaps and follows	A overlaps B	AA_B_AB_BB	B starts during A but ends after A.
B follows and is adjacent to	A meets B	A_AA__AB_BB	The first actions of B is immediately after the last action of A.
B is nearby and follows	A before B	A_A_A_BB_B	The first action of B is a few actions after the last action of A.

Table 3.1: This table shows all the possible context relationship features involving a sequence B. An example and the corresponding Allen relation is given using another sequence A. The underscores in the example are just to demonstrate that the actions in the two sequences may be interrupted by actions from other sequences or actions without sequences.

The last difference between sequence clustering and the other two clustering stages is that is adds a third similarity measurement into the computation of overall similarity. I refer to this as feature similarity. Similar to context similarity, it takes a vector of features for each sequence and computes the similarity using Dice. The difference is that there are only a few features in each vector and they are not context features. I separated this similarity measurement from context similarity because I wanted to weight it differently and not let it get diluted due to the sheer number of elements in a context feature vector. I chose the features based on observation of the golden logs and how the variance in these features was relatively small within a single event compared to the variance of the features between events. There four features are:

1. The number of times a sequence appears in the first third of a log.
2. The number of times a sequence appears in the middle third of a log.
3. The number of times a sequence appears in the last third of a log.
4. The average expansion factor of the sequence.

The three similarity scores are combined into one, with most of the weight going to context similarity and then feature similarity and lastly surface similarity. Clustering then precedes as normal using the matrix of pair-wise similarities.

3.10 System Evaluation

This is the final stage, but it is not really part of the system the way the rest of the stages are. Rather, it is designed to evaluate the performance of all the other stages holistically using the output of the previous two stages. This stage

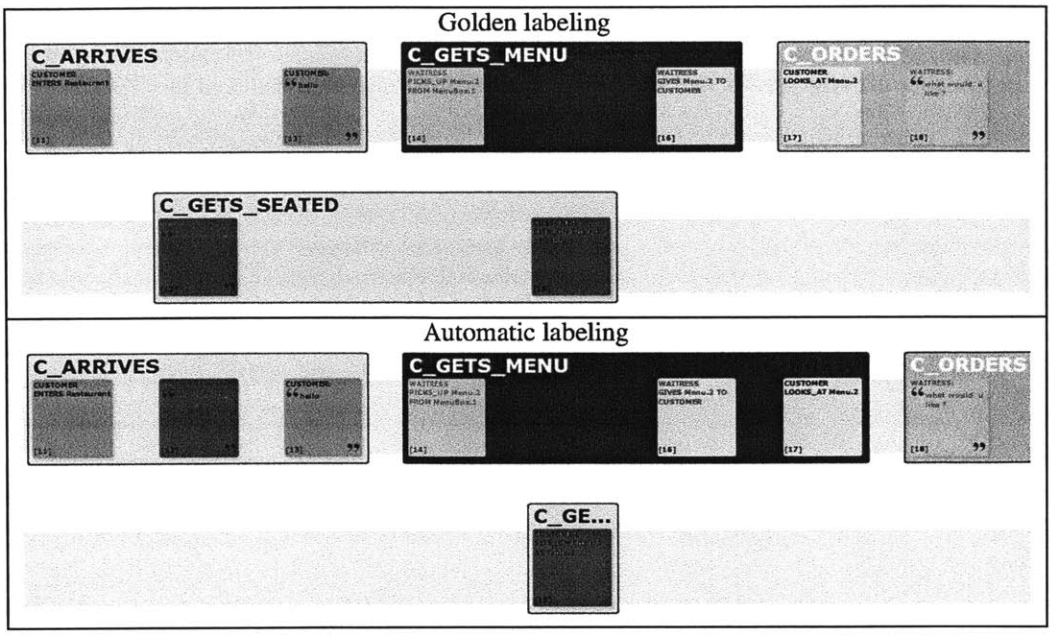
compares the automatic labellings produced in stage 7 with the golden, manual labellings on a subset of the log files. There are several comparisons that take place both intra-log and inter-log and a variety of scores are output to a file.

Both the intra-log and inter-log comparisons are based on the idea that assigning actions and dialogue lines to event-sequences is a similar task to assigning elements to clusters. Several different extrinsic clustering evaluation measures are used to evaluate how well the event-sequence assignments correspond between the golden labeling and the automatic labeling. The intra-log measurements evaluate whether elements that appear in the same event-sequence instance in the golden labeling also appear in the same event-sequences instance in the automatic labeling. The inter-log measurements evaluate whether elements in golden event-sequences corresponding to the same event, are also in automatic event-sequences corresponding to the same automatic event (i.e. event-sequence cluster). See figure 3.8 for pictorial examples. One thing to note is that if human annotators were just given logs to label with events, they may have produced many of the same erros my system produces. However, they are also given examples of each event label, which makes the task easier.

For intra-log evaluation, golden event-sequences are treated as the classes, automatic event-sequences are treated as the clusters and actions and dialogue lines are treated as the elements. Then several information-theoretic extrinsic measures are taken and output to a file, including V [59], NVI [60] and B-cubed [61]. V measures homogeneity (i.e. to what extent do the clusters only contain elements from the same class) and completeness (i.e. to what extent are elements of the same class assigned to the same cluster) using the entropies and conditional entropies of the classes and clusters and combine them by taking the weighted harmonic mean. NVI is a normalized version of the earlier VI (Variation of Information) which also uses the conditional entropies to give a measure reflecting both the completeness and homogeneity of the clustering. B-cubed uses the notion of precision and recall with respect to a particular reference element and averages these over all the reference elements. A total B-cubed score can be obtained by taking the harmonic mean of b-cubed precision and recall.

Each metric has its own advantages and disadvantages. V is nice because it gives values for homogeneity and completeness independently allowing you to weight one or the other more depending on what is more important in a given application [59], but it does not satisfy the metric axioms [60]. All of them lie on a scale from 0 to 1, with values that do not depend on the dataset size directly [59, 60, 61]. Only B-cubed satisfies the rag-bag criteria as defined in [62].

The rag-bag criteria is the idea that a clustering which includes one large cluster with many small elements from tiny classes is better than scattering those tiny class elements throughout other clusters. This is a useful criteria for evaluating the system because it is better to have one cluster (the other / not in a sequence cluster) to contain all the actions from golden event-sequences that were missed in automatic event-sequences than to have them stuck in other automatic event-sequences.



(a) Intra-log example. The orange and pink pair of actions show similarities which we want to reward through intra-log evaluation, the yellow and purple pairs show differences we want to penalize. The orange pair of actions demonstrates being in the same sequence in both labellings. The pink pair demonstrates being in different sequences in both labellings. The purple pair is in the same sequence in the top labeling but not in the bottom and the yellow pair is the opposite.

(b) Inter-log example. The orange, pink, magenta, and gray pairs of actions demonstrate having the same event label in both labellings. Note it doesn't matter if the event labels are the same between the golden and automatic labeling, just within each labeling as in the orange case. This is because the automatic labeling doesn't have the same set of events, just event-sequence clusters. The cyan and green pairs demonstrate another similarity to be rewarded, having different events in both labellings. The purple and yellow pairs demonstrate differences to be penalized. Purple shows having the same golden events but different automatic events. Yellow shows having the same automatic events but different golden events.

Figure 3.8: This shows examples of the sort of similarities and differences that are being evaluated between the golden labeling and automatic labeling both intra-log and inter-log.

These information theoretic approaches also are more suitable than many other approaches (such as matching) because they are less affected by the number of clusters. This is a useful criteria for evaluating the system because the automatic labeling system only labels the frequent event-sequences, where as the golden labellings cover all [2] event-sequences in a log. This creates a situation where there will be much fewer "clusters" than "classes" and one cluster will have contain a large portion of the elements.

[2]Based on a human determined list of events and according to a human's discretion.

For inter-log evaluation the same measures are used, and actions are still considered the elements. However, this time the golden events, rather than the specific golden event-sequence instances are considered the class labels. Similarly the automatic event-sequence clusters are considered the cluster labels rather than particular sequences instances.

Chapter 4

Analysis, Discussion and Results

This chapter evaluates the performance of the event-discovery system and its components. Although only the system as a whole can currently be quantitatively evaluated, the most important components are analyzed qualitatively. I discuss and give examples of what the components do well and what they do poorly and my explanations for their behavior. Chapter five details some of my ideas for future work that could be done to improve the performance and accuracy flaws I have noticed. This chapter also elaborates on the parameters of the system and how they were tuned in response to component outputs in order to achieve the best results. The first four sections focus on the key components of the system, word clustering, dialogue and action clustering, event-sequence mining and event-sequence clustering. The final section analyzes the overall labeling performance of the system, both qualitatively and quantitatively. All the components of the system are deterministic so the results discussed are based on the best results achieved using the parameters given in appendix B unless otherwise noted. Appendix A has sample output from the various components that was produced using the parameters given in Appendix B.

4.1 Word Clustering

The word clustering stage received a lot of attention in terms of experimentation, research and refinement. The end result was a clustering component capable of accurately clustering the vast majority of frequently used words and many of the less frequently used words. Though I have no quantitative evidence, the clustering performed quite well with respect to its purpose in the event-sequence discovery system, which I will demonstrate with examples. By this I mean that the clustering does not perform as well as it could if its purpose was just to cluster the words semantically. If that were the goal, the result would be very small clusters of synonyms.

The ultimate goal of this stage is to combine words together that are used similarly in the context of restaurant dialogue. In fact, the clustering parameters were purposefully tuned to have fewer broader clusters rather than smaller more semantically accurate clusters (e.g. the preference factor and the context similarity factor etc. see section B.1). This resulted in clusters such as adjectives that were used similarly (e.g. used to describe food, used to describe a person etc.) were clustered together even if they were antonyms (see "great" cluster in table A.1 for example). This seems to be more helpful for the dialogue clustering process than the alternative as discussed more in the following section.

The system was really good at keeping clusters consistent in terms of part of speech type (e.g. noun, verb, determiner etc.), and to a lesser extent form (e.g. number, tense etc.). This makes sense as narrow contexts are the primary component of measuring word similarity, and word types and forms vary greatly in the narrow contexts they appear in. For example, adjectives appear before nouns, noun verb noun is a very common structure etc. some example, word type clusters are shown in figure 4.1. As described in section 3.3, the affinity propagation clustering method used for clustering in the event discovery system, automatically produces an exemplar for each cluster it outputs. This exemplar represent the word, action, dialogue line or sequence within the cluster that is most representative of the clsuter as a whole. Each of the clusters shown in this section and the section on dialogue and action clustering are labelled with their exemplar.

ask	had	put	use
asked	hate	said	wanted
ate	live	sat	work
came	love	see	write
care	made	seem	
come	make	stole	
forgot	ordered	took	

(a) ate

alright	handsome	strange
awesome	hawt	sweet
awful	interesting	weird
cute	nasty	welcome
fantastic	perfect	yummy
fast	popular	

(b) awesome

Figure 4.1: Two sample clusters of words of generally the same type. The first is mostly past-tense verbs and the second is adjectives.

The system is also good at its primary goal of grouping together words that are used similarly in the restaurant context. For example, adjectives that describe the customer's opinion of the food (figure 4.1b), parts of the restaurant ("corner" cluster in table A.1), etc. In a few cases it performed better with respect to its goal of helping to semantically cluster dialogue lines than a human probably would have. For example, there were some groups of words that I initially thought were odd to cluster together. However, as I worked with the dialogue more I realized that they were being used in very similar ways in the context of a restaurant. For example, "house," "red," and "white" were all clustered together and I realized it was because they were all used in conjunction with the word "wine" (figure 4.2). Another good example is "enjoy" and "heres," which are both used by the waitress in contexts like "Enjoy your salad sir."

One problem with the word clustering system is that it treats words that belong to the same phrase separately, such as

filet	house	red	whit	white

Figure 4.2: The cluster of words with "white" as the exemplar.

berry	nactarine	nectarine	tangerine
cherry	nacterine	nectartine	
cobb	nectar	necterine	
grilled	nectarin	nectorine	

Figure 4.3: The cluster of words with "nectarine" as exemplar

"fillet" and "mingon." However, it does succeed for the most part at clustering together the first words of food phrases (e.g. "cobb" from "cobb salad," "nectarine" from "nectarine tart," "grilled" from "grilled salmon," etc. as seen in figure 4.3).

Another potential problem with the word clustering system is that it sometimes forms multiple clusters corresponding to the same category of words. For example, geographic locations, people's names, numbers etc. In some cases, like geographic locations and people's names, these should definitely be combined into one cluster. The likely problem is that most words of these types are used infrequently and there are several different contexts they may be used in, so they tend to be grouped according to the particular context they occur in. In other words, they don't have a sufficient context distribution to combine the separate groups. Examples of these are shown in figure 4.4.In other cases, the sub-groups make more sense and might actually be useful because they are separated by slight variations in the way the words of the category are used. For example, the "sixteen" cluster shown in table A.1 has a lot of numbers that were probably used to refer to ages mostly.

There are two main sets of words the system clusters poorly. Both are clustered poorly primarily because they lack enough context information to be clustered effectively by this context-focused system. The first is infrequent words, often in the form of typos or misspellings. These either wind up as random words in otherwise logical clusters or in large clusters of random words. Table A.1 shows some examples of both. The problem is with only a few context examples to work with, the system is usually unable to get a good enough context distribution to determine how the word is typically used and group it with similar words. The other problem-causing group is words that tend to occur in very small utterances, for example, "hello," "thanks," "yes," and the other words shown in figure 4.5. The problem

belgium	england	maine
boston	france	san
cambridge	germany	spain
carolina	holland	texas

(a) germany

alberta	czech	iceland	portugal
australia	europe	ireland	stranger
brasil	greece	italy	
brazil	hector	nj	

(b) portugal

Figure 4.4: Sample word clusters corresponding to geographical locations.

bye	hey	lol	ok	sure	well
haha	hi	no	okay	thanks	yeah
hello	how	oh	sorry	thankyou	yes

Figure 4.5: The cluster of words with "hello" as exemplar

here is not that there is an insufficient amount of context information, its just that most of the context isn't useful. It doesn't impart any knowledge about how the word is used in dialogue beyond the fact that it is used by itself. Both of these problems could be fixed by setting aside words that meet these criteria to be clustered manually. Alternatively, in the first case, simply removing the least frequent words might be a good solution.

Most of the parameters used by the word clustering stage were empirically determined through trial and error to balance between accuracy and efficiency. Some were tuned to produce the larger, usage-similarity based clusters as well. A description of all the parameters and the values used to produce the results described in this thesis are given in section B.1.

4.2 Dialogue and Action Clustering

The purpose of dialogue and action clustering is to reduce the number of unique elements in the game play logs, thereby allowing patterns to be found more easily. The goal is to group dialogue lines and actions by the way they are used, not their meaning. The difference between meaning and usage is apparent in the simple example of "yes" and "no." They have opposite meanings, but can both be used to respond to the same questions, so they should be grouped together in this stage. Any important differences in meaning will be captured by variance in subsequent action and dialogue. If the customer says "no" to "Would you like another beer?" the waitress will perform a different set of actions than if the customer had said "yes." The idea is that the specific objects of actions, and detailed meanings of dialogue lines, are not needed to determine event labels, rather the types of objects and intent of dialogue lines are. Although, using the specific objects might be useful in labeling event instances (see section 5.2.4).

With that goal in mind, the dialogue and action clustering system performs well. The action clustering in particular worked very well. As mentioned before (section 3.2), some action clustering was done in the initial log processing. This grouped actions by clustering some of the in game objects based on similarities in the way they were used. However, many actions that serve the same purpose are still separated. Some examples include: different variations of the customer looking at the menu (e.g. while holding it or while it is on the table); multiple ways the waitress can give the menu to the customer (e.g. putting it on the table in front of him or handing it to him) etc. The action clustering stage does a good job of grouping together these different variations of accomplishing the same goal. For example,

table 4.1 shows a cluster of actions relating to the customer receiving the menu.

Waitress give menu					
Actor	Action	Object	Precondition	Effect	Count
Waitress	Get off	Fruit bowl	Waitress sitting on fruit bowl, customer holding fruit bowl	Waitress standing	1
Waitress	Get off	Menu	Waitress sitting on held menu	Waitress standing	3
Waitress	Give	Menu	Waitress standing holding menu	Customer holding menu	2255
Customer	Give	Menu	Customer sitting holding menu	Waitress holding menu	585
Customer	Give	Menu	Customer standing holding menu	Waitress holding menu	229
Customer	Look at	Menu	Customer sitting holding menu		2545
Waitress	Look at	Menu	Waitress standing holding menu		678
Customer	Look at	Menu	Customer standing holding menu		329
Waitress	Look at	Menu	Waitress standing, menu on table		145
Customer	Look at	Menu	Customer sitting, menu on table		1086
Waitress	Look at	Menu	Waitress standing, customer holding menu		69
Customer	Look at	Menu	Customer sitting, menu on CHAIR		93
Customer	Look at	Menu	Customer sitting, waitress holding menu		134
Waitress	Pickup	Menu	Waitress standing, menu in menu box	Waitress holding menu	3966
Customer	Pickup	Menu	Customer sitting, menu on table	Customer holding menu	950
Waitress	Pickup	Menu	Waitress standing, menu on table	Waitress holding menu	1824
Waitress	Pickup	Menu	Waitress standing, customer holding menu	Waitress holding menu	492
Customer	Pickup	Menu	Customer sitting, menu on CHAIR	Customer holding menu	94
Customer	Pickup	Menu	Customer sitting, waitress holding menu	Customer holding menu	331
Customer	Pickup	Menu	Customer standing, waitress holding menu	Customer holding menu	117
Customer	Pickup	Pan	Customer sitting on pot, pan on stove	Customer holding pan	1
Customer	Put down	Menu	Customer sitting holding menu	Menu on table	1895
Waitress	Put down	Menu	Waitress standing holding menu	Menu on menu box	2503
Waitress	Put down	Menu	Waitress standing holding menu	Menu on table	1482
Waitress	Put down	Menu	Waitress standing holding menu	Menu on podium	219
Waitress	Put down	Flower	Waitress sitting on flower holding flower	Flower on trash	1
Waitress	Put down	Menu	Waitress standing holding menu	Menu on floor	200
Waitress	Put down	Menu	Waitress standing holding menu	Menu on CHAIR	235
Waitress	Put down	Menu	Waitress standing holding menu	Menu on register	25
Waitress	Put down	Menu	Waitress standing holding menu	Menu on menu	6
Customer	Put down	Menu	Customer standing holding menu	Menu on menu	6
Customer	Put down	Menu	Customer standing holding menu	Menu on fruit	1
	Spawn	Menu			4131

Table 4.1: A sample output action cluster relating to the waitress giving a menu to the customer. Note, if we implement pruning of rare actions, many of the inappropriate actions in the cluster above would no longer be present.

The actions in the original lexicon are also defined by the state of the objects and actors involved, such as how many bites have been taken out of the food, the location of the object and actor etc. The action clustering system does a good job of combining actions that only differ by states unimportant to the purpose of the action. For example, in the cluster shown in table 4.2, the customer eating food with no bites taken is clustered with eating food with one or two bites taken and eating held food is clustered with eating food on the table among others.

Table 4.2: A sample output action cluster relating to the customer eating food and drinking beverages.

Customer eat FOOD					
Actor	Action	Object	Precondition	Effect	Count
	Delete	Register	Customer holding register	Register deleted	7
Customer	Eat	BEVERAGE	Customer sitting, BEVERAGE on table	BEVERAGE sipped	3082
Customer	Eat	BEVERAGE	Customer sitting, BEVERAGE on table, sipped once	BEVERAGE sipped	2979

continued on next page ...

Actor	Action	Object	Precondition	Effect	Count
Customer	Eat	BEVERAGE	Customer sitting, BEVERAGE on table, sipped twice	BEVERAGE sipped	2685
Customer	Eat	OLD FOOD	Customer sitting, OLD FOOD on table	OLD FOOD bitten	767
Customer	Eat	OLD FOOD	Customer sitting, OLD FOOD on table, bitten once	OLD FOOD bitten	778
Customer	Eat	OLD FOOD	Customer sitting, OLD FOOD on table, bitten twice	OLD FOOD bitten	770
Customer	Eat	FOOD	Customer sitting, FOOD on table	FOOD bitten	3479
Customer	Eat	FOOD	Customer sitting, FOOD on table, bitten once	FOOD bitten	3463
Customer	Eat	FOOD	Customer sitting, FOOD on table, bitten twice	FOOD bitten	3406
Customer	Eat	BEVERAGE	Customer sitting holding BEVERAGE	BEVERAGE sipped	920
Customer	Eat	BEVERAGE	Customer sitting holding BEVERAGE, sipped once	BEVERAGE sipped	767
Customer	Eat	BEVERAGE	Customer sitting holding BEVERAGE, sipped twice	BEVERAGE sipped	645
Customer	Eat	FOOD	Customer standing, FOOD on table, bitten twice	FOOD bitten	92
Customer	Eat	FOOD	Customer standing, FOOD on chair	FOOD bitten	2
Customer	Eat	DIRTY DISH	Customer sitting, DIRTY DISH on table		1258
Customer	Eat	WINE	Customer sitting, WINE on table	WINE sipped	175
Customer	Eat	DIRTY DISH	Customer sitting holding DIRTY DISH		180
Customer	Eat	Flower	Customer sitting, flower on table	Flower bitten	893
Customer	Eat	Table	Customer sitting		371
Waitress	Eat	Flower	Waitress standing, flower on table	Flower eaten	340
Customer	Eat	Menu	Customer sitting, menu on table		150
Customer	Eat	Vase	Customer sitting, vase on table		213
Customer	Eat	WINE	Customer sitting, WINE on OLD FOOD	WINE sipped	3
Customer	Eat	Waitress	Customer sitting, Waitress standing		184
Waitress	Eat	DIRTY DISH	Waitress standing, DIRTY DISH on table		81
Waitress	Eat	BEVERAGE	Waitress standing, BEVERAGE on floor, sipped twice	BEVERAGE sipped	1
Waitress	Eat	FOOD	Waitress sitting, FOOD on ground, bitten twice	FOOD bitten	1
Customer	Eat	DIRTY DISH	Customer sitting on BEVERAGE holding DIRTY DISH		1
Customer	Eat	BEVERAGE	Customer sitting, BEVERAGE on floor, sipped twice	BEVERAGE sipped	1
Customer	Eat	WINE	Customer standing, WINE on blender, sipped twice	WINE sipped	1
Customer	Eat	FOOD	Customer sitting on bartender holding FOOD, FOOD bitten once	FOOD bitten	4
Customer	Eat	FOOD	Customer sitting on bartender holding FOOD, FOOD bitten twice	FOOD bitten	4
Waitress	Eat	FOOD	Waitress sitting on FOOD, FOOD on counter	FOOD bitten	2
Customer	Eat	Trash	Customer sitting		1
Waitress	Eat	FOOD	Waitress standing, FOOD on OLD FOOD, bitten once	FOOD bitten	1
Customer	Eat	Menu	Customer sitting, menu on fruit		1
Customer	Eat	Waitress	Customer sitting on counter, waitress sitting on FOOD		1
Waitress	Get off	Register	Waitress sitting on register on table	Waitress standing	1
Customer	Get off	Flower	Customer sitting on flower on podium	Customer standing	1
Customer	Look at	FOOD	Customer sitting, FOOD on table		160
Customer	Look at	BEVERAGE	Customer sitting, BEVERAGE on table		86
Customer	Look at	Flower	Customer sitting, flower on table		253
Customer	Look at	DIRTY DISH	Customer sitting, DIRTY DISH on table		76
Customer	Pay	Bill	Customer standing, bill on OLD FOOD		1
Customer	Pickup	Flower	Customer sitting, flower on table	Customer holding flower	476
Customer	Pickup	FOOD	Customer sitting, FOOD on table	Customer holding FOOD	652
Customer	Pickup	BEVERAGE	Customer sitting, BEVERAGE on table, sipped once	Customer holding BEVERAGE	317

continued on next page ...

Actor	Action	Object	Precondition	Effect	Count
Customer	Pickup	BEVERAGE	Customer sitting, BEVERAGE on table, sipped twice	Customer holding BEVERAGE	187
Waitress	Pickup	Trash	Waitress sitting on fridge, trash on fridge	Waitress holding trash	1
Customer	Pickup	WINE	Customer standing, WINE outside	Customer holding WINE	1
Waitress	Pickup	FOOD	Waitress standing, FOOD on floor, bitten twice	Waitress holding FOOD	6
Customer	Pickup	BEVERAGE	Customer sitting, BEVERAGE on floor, sipped twice	Customer holding BEVERAGE	1
Waitress	Pickup	Trash	Waitress standing, trash on food	Waitress holding trash	1
Customer	Pickup	Trash	Customer sitting, waitress holding trash	Customer holding trash	1
Customer	Put down	FOOD	Customer sitting holding FOOD, bitten once	FOOD on table	107
Customer	Put down	BEVERAGE	Customer sitting holding BEVERAGE	BEVERAGE on table	518
Customer	Put down	BEVERAGE	Customer sitting holding BEVERAGE, sipped once	BEVERAGE on table	451
Customer	Put down	BEVERAGE	Customer sitting holding BEVERAGE, sipped twice	BEVERAGE on table	286
Customer	Put down	DIRTY DISH	Customer sitting holding DIRTY DISH	DIRTY DISH on table	835
Customer	Put down	FOOD	Customer sitting holding FOOD	FOOD on table	890
Customer	Put down	Flower	Customer sitting holding flower	Flower on table	250
Customer	Put down	Fruit	Customer standing holding fruit	Fruit on trash	1
Customer	Put down	WINE	Customer standing holding WINE	WINE on bar	4
Customer	Touch	BEVERAGE	Customer sitting, BEVERAGE on table		129
Customer	Touch	Waitress	Customer standing, waitress standing		353
Customer	Touch	Flower	Customer sitting, flower on table		54
Customer	Touch	Table	Customer sitting		57

Action clustering isn't perfect, it often mis-classifies actions that occur very infrequently. For example, we see some random actions in both table 4.1 and table 4.2, including the waitress putting down flowers she is sitting on and the customer paying a bill that is on top of food while standing. This is because only context information is being used to cluster the actions. If other features were added into the similarity computation, such as the objects, verbs and states associated with the actions. We could probably improve the performance for these infrequent actions, and probably action clustering in general. Modifying this stage to do such action clustering first and then following up with context and surface similarity based dialogue clustering would be an interesting approach for future work. However, most rare actions are bizarre, like picking up the trash while sitting on the refrigerator, and it is not that important to get them correct for the purpose of event discovery.

Another mistake the action clustering system makes is to sometimes cluster together actions that occur at similar points in a game, but don't correspond exactly to the same goal. For example, in table 4.1, we see that actions relating to the customer getting the menu are clustered with actions corresponding to the customer looking at the menu, and the waitress putting away the menu. This type of error is likely because of the very limit context (one, element to either side) used to compute context similarities for the dialogue and actions. This was chosen as a necessity to keep the stage from consuming too much memory and being too slow. Some efficiency improvements, as discussed in section 5.2.1, should allow us to use larger contexts and resolve this issue. In practice this does not seem to cause too many problems, because it is not that common and because the actions that are clustered together because of this are

can i get a beer ?	can i have some water ?
how can i help you ?	how may i serve you ?
coffie please	beer please
coffe please	beer please
some water please	a beer please
spaghetti please	steak please
cheese pie please	filet minon please
white wine	red wine
comin right up	coming right up
may i get you something to drink ?	can i get u anything to drink ?
right up	right away
here is your beer	here is the wine
house	red
did you need something ?	do you need anything ?

Table 4.3: Example pairs of dialogue lines found to be identical in terms of the sequences of word clusters they are composed of.

typically of the same event.

The dialogue clustering worked well, but was not perfect. The dialogue clustering is done in two stages: first lines that are identical in terms of the sequence of word clusters they are composed of are grouped together; second the resulting groups of unique lines are clustered simultaneously with actions as described in section 3.4. The success of the word clustering directly translated into the success of the first phase of dialogue clustering. A few examples of pairs that were found to be identical by this method are shown below.

The second phase of dialogue clustering is somewhat less successful. The sheer magnitude of possible dialogue lines (about 50,000 unique lines for each character in only 5,000 games) leaves many unusual dialogue lines in clusters to themselves (of about 10,000 total clusters for each character, about 9,000 are individual dialogue lines). A different approach to dialogue or a modified approach in which these unclustered lines are forced into other clusters might help with this situation. On the positive side, the output contains many clusters of dialogue lines, where the purpose is easily identifiable by a human. In most of these cases, the exemplars of the clusters are very representative of this purpose. One good example is the following excerpt from the cluster with the exemplar "thank you."

- **thank you**
  - thank you please
  - thank you yet again
  - thank you dear
  - thank you beautiful
  - thank you p
  - thank you misses
  - thanks but
  - thanks rachel
  - thanks alot
  - thanks toots
  - thanks kindly
  - thanks slave

- thanks for that
- thanks for asking
- thanks a lot
- sorry for the trouble
- thanx a lot
- please and thank you mam
- thank you much
- thank you though
- thank you mam
- thank you darlin
- thank you babe
- thank you darling
- thank you kindly
- thank you slave
- thank you alice
- thank you jenni
- oh thank you mam
- why thank you maam
- hello and thank you
- thanks for a nice meal
- thanks for the fine service
- thanks have a nice day
- thanks have a good one
- ...

One drawback of the way the current system handles dialogue is the inability to stick dialogue lines in multiple clusters. This would be very useful as people frequently use complex dialogue lines to fulfill several purposes simultaneously. For example, the dialogue line "Excuse me, but my table is dirty. Water would be great." both complains about the cleanliness of the table and requests a water. It would be useful to be able to interpret the dialogue as either purpose for mining and labeling sequences. Being able to put dialogue lines in multiple event-sequences in the same game would go hand-in-hand with this. Anoother option might be to have support for chopping up dialogue lines on boundaries found statistically, or based on punctuation. Another challenge with dialogue comes from the fact that the game limits the length of a dialogue line. When people reach the limit they usually continue what they were saying in an immediately following dialogue line. This means that in addition to one dialogue line having multiple purposes, sometimes two dialogue lines combined have one purpose. Analysis of the time between the dialogue lines and the length of the first dialogue line could fix this by combining the two dialogue lines into one.

The dialogue clustering system has many of the same weaknesses that the word clustering system has. It handles many of the most unique dialogue lines poorly, typically placing them in their own single-element cluster. In some cases these lines would be be useful to put in a more correct cluster, but overall these types of mistakes don't affect the performance of the system as a whole to any great degree. This is because these dialogue lines are often unique to the point where they do not belong to any typical event-sequence anyway, so they do not have an impact on our

attempts to discover event-sequences. Some examples of these lines include "I take blue color pills," "I thought you were exploded," and "Here I come dinosaurs." If we wanted to capture sequences corresponding to more unusual events, or less typical sequences for normal events then we would have to resolve this issue.

Another challenge faced by the dialogue system is that the ideal solution contains a few very large clusters corresponding to the most common types of restaurant dialogue such as "I'll have the ????," and many small clusters corresponding to less common things such as flirting and talking about the game itself. When choosing the parameters, particularly the preference parameter (see section B.2), I had to make a trade-off between fewer large clusters, which would group many of the small clusters together, and more small clusters, which would split up some of the larger clusters. It was basically a trade-off between homogeneity and completeness of the clustering. I found for my purposes that leaning toward fewer, larger clusters produced the best results in the following stages of the system. More examples of partial sample output clusters (most clusters were too large to list in their entirety) are given in appendix A section A.2.3.

4.3 Event-Sequence Mining

The sequence mining and filtering system works pretty well. It does a very good job of finding the most typical event-sequences, but does not find all the event-sequences. It also finds a fair number of non-event-sequences, which may be incomplete event-sequences, combinations of parts of different event-sequences, or often silly behaviors on the part of the players that somehow occurred frequently enough to get mined. The sequence mining and filtering stage is efficient, taking at most 10s of minutes to run on a several year-old laptop. Furthermore, it finds a reasonable number of candidate event-sequences, on the order of a few thousand depending on the exact parameters. Some typical event-sequences that it finds are shown below.

One of the most common types of errors the system makes is to find incomplete event-sequences. Even with the pseudo-closed-sequence filtering in place (see section 3.7), many incomplete sequences are mined. Sequences that are missing elements in the middle rather than at either end will not get caught by the pseudo-closed sequences filtering. In addition, sometimes different event-sequence instances have a common sub-sequence but vary greatly in their other components, so that even if they are mined in their entirety, the common sub-sequence will also be mined separately. Some examples of incomplete sequences are shown in figure 4.7.

Another common error is mining candidate sequences that overlap multiple events, as defined by Jeff's previous work (see section 2.5). For example, candidate sequence 1 shown below corresponds to parts of both the customer entering the restaurant event and the customer getting seated event. The second sequence has parts of both the waitress serving

- Waitress picks up food from counter - Waitress puts down food on table - Waitress says "enjoy sir"
- Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table - Waitress says "here is your meal"
- Customer says "could i have the bill please" - Waitress touches register - Waitress picks up bill - Waitress puts down bill on table - Customer picks up bill
- Customer says "could i have the bill please" - Waitress touches register - Waitress picks up bill - Waitress puts down bill on table - Customer looks at bill
- Waitress says "what would you like" - Customer says "spaghetti marinara" - Waitress picks up menu from table - Waitress puts down menu on menu box - Delete menu
- Customer looks at menu - Customer says "spaghetti marinara" - Waitress says "ok coming right up"
- Customer looks at menu - Customer puts down menu - Customer says "spaghetti marinara"
- Waitress says "hello sir" - Customer says "hello" - Waitress picks up menu
- Customer says "hello" - Waitress says "table for one"

Figure 4.6: Examples of good candidate event-sequences mined by the system.

- Waitress puts down food on table - Waitress says "here is your meal"
- Waitress touches register - Waitress picks up bill
- Waitress puts down bill on table - Customer says "thank you"
- Waitress gives food to customer - Customer puts down food on table

Figure 4.7: Examples of candidate event-sequences that correspond to partial actual event-sequences.

food and the customer eating events. This is not necessarily a big issue, as these events were defined subjectively by a single person, and may not be the only way to divide up the steps of a typical restaurant interaction.

1. Waitress says "hello" - Customer says "table for one please" - Waitress says "follow me" - Customer sits down
2. Waitress puts down food on table - Waitress says "here is your meal" - Customer eats food - Customer eats food - Customer eats food

One other mistake the system makes is finding a fair number of candidate sequences that basically correspond to the waitress and or customer doing silly and random things. These are sequences that occur just frequently enough to be mined by the system, and are close together when they do appear in order to not be filtered out based on the expansion factor (section 3.6). The majority are composed of just two actions as shown below. Having a higher minimum frequency count for shorter sequences could help prevent this. Another option might be to remove all the silly actions from the logs before mining. However, we do want the final A.I. system to be able reproduce unusually behavior as well as typical behavior.

- Waitress sits on vase - Customer touches fruit
- Customer gives menu - Customer gives vase
- Waitress eats trash - Customer puts down trash
- Waitress puts down pan - Customer puts down dirty dish
- Customer eats fruit - Customer picks up bill
- Customer gets off wine - Waitress cleans dirty dish - Delete dirty dish - Customer looks at menu

Figure 4.8: A cluster of unusual candidate event-sequences mined by the system.

4.4 Event-Sequence Clustering

Given the event-sequences that are mined, the event-sequence clustering system works very well. Most of the clusters are consistent enough to be identified as a particular event by a human. The excerpt below is from a cluster I identified as corresponding to the event of the waitress serving food.

Most of these good clusters also contain a few mistakes, such as the ???? sequence in the cluster above. However, most of the strange sequences mentioned in the previous section tend to wind up in their own clusters. About 10-15 percent of the clusters are primarily strange sequences like in figure 4.8. The fact that candidate sequences might be partial or have components of multiple events also messes things up a little bit. Some of the events that Orkin defined are not represented by their own mostly uniform cluster. For example, the customer getting a menu is split between clusters mostly related to the customer arriving, getting seated and ordering, as seen in the example clusters in section A.3.

4.5 Overall Performance

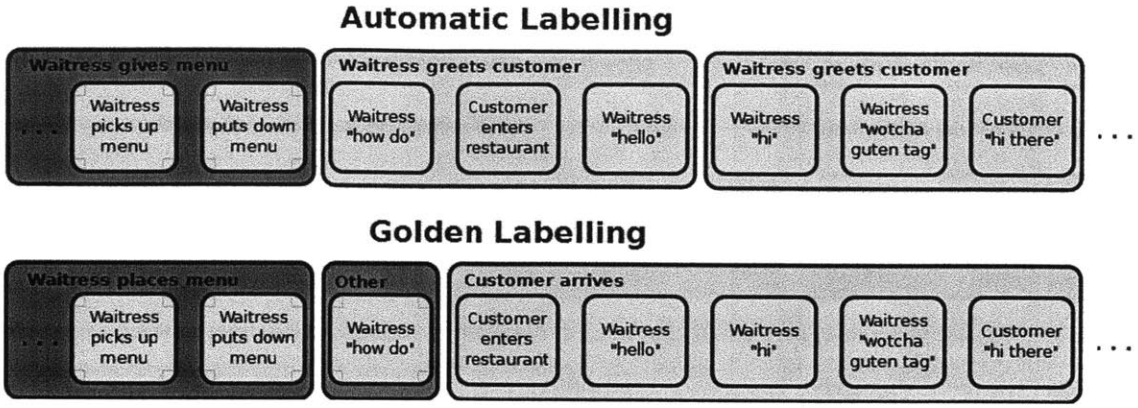
The evaluation of the system overall is difficult to analyze, as I can only readily compare the performance of the system to itself using different parameters. That being said, looking at how the overall performance metric varies with changes to the system is interesting. Unfortunately I have so far only been able to do this to a limited degree due to the time required to run the system from beginning to end. It is also interesting to look at particular logs that the system scores well on and particular logs that it scores poorly on to see what leads to each case.

As described in section 3.10, a variety of extrinsic clustering measures were used to evaluate the system by comparing how it assigned game-actions to event-sequences relative to the gold-standard of Jeff's manually labeled logs. Each of the metrics behaved slightly differently, but the one that seemed to correspond most accurately with actual performance[1] for this application was b-cubed [61], so I will be discussing the performance of the system in terms of that measure.

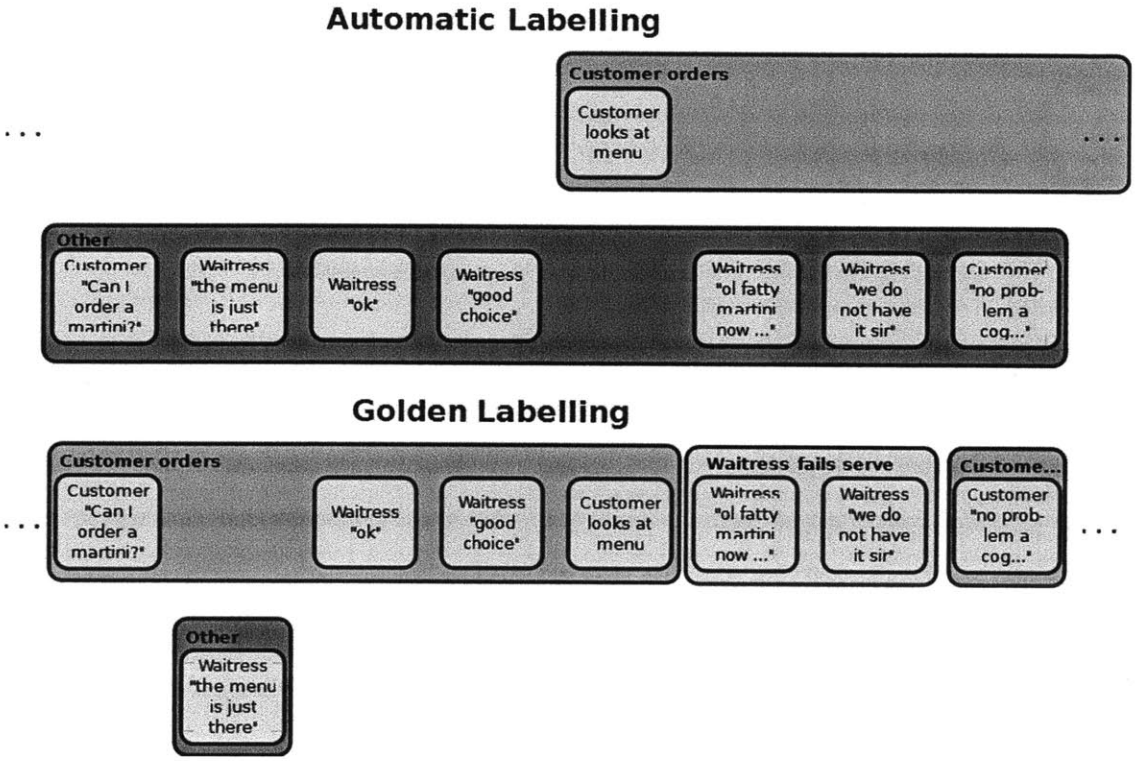
[1] Based on empirical qualitative evaluation.

- **Waitress serves customer food**
  - Waitress puts down food on table - Waitress says "here is your meal"
  - Waitress puts down food on table - Waitress says "here is your meal" - Customer picks up food
  - Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table
  - Waitress puts down food on table - Waitress says "here is your meal" - Customer eats food - Customer eats food - Customer eats food
  - Waitress puts down food on table - Customer says "thank you" - Customer eats food - Customer eats food - Customer eats food
  - Customer says "where is my salad" - Waitress says "ok coming right up" - Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table
  - Waitress puts down food on table - Waitress says "this is on the house" - Customer eats food - Customer eats food
  - Waitress gives blender to customer - Customer eats flowers
  - Waitress puts down food on table - Customer says "this looks good" - Customer eats food
  - Waitress looks at fridge - Customer eats trash
  - Waitress puts down food on table - Waitress says "that is filet mingon" - Customer eats food
  - Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table - Waitress says "here is your meal"
  - Waitress puts down food on table - Waitress says "anything else sir"
  - Waitress puts down food on table - Waitress says "anything else sir" - Customer looks at menu
  - Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table - Customer eats
  - Waitress puts down food on table - Customer picks up food - Customer eats food - Customer eats register
  - Waitress picks up food from counter - Waitress puts down food on table - Waitress says "enjoy sir"
  - Waitress puts down food on table - Waitress says "enjoy sir" - Customer picks up food - Customer eats food - Customer eats food - Customer eats food
  - Waitress puts down food on table - Waitress says "enjoy sir" - Customer eats food
  - Waitress gives food to customer - Customer puts down food on table
  - Waitress puts down food on table - Customer says "thank you" - Customer picks up food - Customer eats food - Customer eats food
  - Waitress picks up food from counter - Waitress puts down food on table
  - ...

Figure 4.9: A sample event-sequence cluster, predominantly corresponding to the waitress serving food event.



(a) Best log excerpt



(b) Worst log excerpt

Figure 4.10: Excerpts from the best and worst log labellings. In the case of the automatic labeling, the labels on the selected sequences are descriptions of the corresponding sequence clusters.

First, to give a basis of comparison, the best scores I obtained for the system overall were 0.22 for the average intra-log comparison score and 0.31 for the inter-log comparison score (see section 3.10 for more information on how these two scores are produced). Within the labeling corresponding to those best scores, the highest single intra-log score was 0.57, while the lowest was 0.024. Excerpts from the best and worst scoring logs are shown in figure 4.10, along with the corresponding gold standard labeling.

As can be seen in the figure, and not surprisingly, the system performs best with shorter sequences that are not

	Greedy	Optimal
Average intra-log score	0.21	0.22
Inter-log score	0.29	0.31

Table 4.4: The performance of different labeling methods.

Percentage covered factor	Average intra-log score	Inter-log score
0.5	0.18	0.30
0.75	0.21	0.32
0.95	0.22	0.31
1.0	0.20	0.30

Table 4.5: System performance versus percentage covered factor.

interrupted. It also helps when all or most of the elements of the sequences are unique to that sequence cluster. For example, the waitress saying "hello" is almost entirely within the waitress greeting the customer cluster, and picking up and putting down the menu is almost exclusive to the waitress getting the menu cluster. The system's major issue is not finding all the possible sequences, identifying between 3 and 10 sequences in most logs. The worst case log shown above, was particularly sparse and the sequences found did not match up very well with any of the sequences in the golden labeling. This log probably performs poorly because it has more abnormal dialogue and actions than many of those tested. All the golden-labeled logs represent fairly typical restaurant behavior so we don't have any for comparison that are completely bizarre to see how well the system performs quantitatively. However, from qualitative inspection it does not do very well, but no worse than the worst case scenarios that we do test.

I noticed several interesting things while tweaking the system and evaluating the effect on performance. One was that using a greedy approach (based on Orkin's RUN algorithm [17]) to labeling rather than my current "optimal" approach did not lower the performance by a significant amount. On some logs the system did perform worse, but on others it performed the same or a little better. This may be due in part to the fact that the scoring method used in my "optimal" labeling approach is quite simplistic. Also, we have noticed that some game-actions are strongly associate with particular events, which causes greedy approaches to be more effective. A summary of the results when varying the method of labeling are shown in table 4.4.

In addition to the way labeling was performed, I also evaluated the scoring method used in my optimal labeling approach. I varied the parameter controlling the ratio of the coverage component of the score and the normality component of the score. I found that coverage was the most important of the two factors, but that including the normality a little bit did improve the performance slightly. The results of this experiment are summarized in table 4.5. Although as long as some coverage score was included, the performance did not vary greatly with what factor of the score was from coverage.

The time needed to run the system prevented me from doing any performance-metric-based experiments involving,

minimum occurence count \ maximum expansion factor	1.7	2.0	2.3
5	0.22,0.31	0.23,0.32	0.25,0.31
10	0.17,0.31	0.15,0.30	0.23,0.32
15	0.12,0.28	0.15,0.29	0.14,0.30
20	0.09,0.25	0.09,0.26	0.11,0.28

Table 4.6: System performance versus minimum occurrence and maximum expansion factor. The first number is the average intra-log score and the second number is the inter-log score.

Preference factor	**Average intra-log score**	**Inter-log score**
0.6	0.16	0.29
0.8	0.18	0.31
1.0	0.22	0.31
1.2	0.19	0.32

Table 4.7: System performance versus sequence clustering preference factor.

word, dialogue and action clustering as of yet. However, I was able to do a little bit of experimenting with the parameters controlling both sequence mining and sequence clustering. The results are summarized in tables 4.6 and 4.7. It is not entirely clear what the results mean because of the effects of the labeling process on the overall score, but they are somewhat interesting.

It appears that in general, modifying the filtering and mining parameters in order to allow more sequences through improves the performance. This is probably because, since the system is only able to label frequent sequences, many logs are sparsely labeled. Allowing more sequences through improves the amount of the logs that are labeled, improving the overall performance. Also it looks like there is a sweet spot between very few large clusters and many small clusters where the system is able to perform best, which intuitively makes sense. Even though increasing the maximum expansion factor increased the score to a point, it is mostly just due to a denser labeling, and qualitatively seemed worse because many of the new sequences did not seem to correspond well to events. This is an example of how it is difficult to quantitatively evaluate the system because of the complex interactions of the different parts.

Chapter 5

Conclusion

5.1 Contribution

This thesis makes a lot of minor contributions, but its primary contribution is the entire novel event-sequence discovery system which tackles an interesting new problem that has not been attempted before. That is the problem of mining patterns from multi-modal data from human-human interactions, in our specific case a mix of physical actions and natural language input in a virtual world. With the increasing amount of online social interaction between human, these techniques may be applicable to a wide-variety of related problems in the future. The event-sequence discovery system combines algorithms from many different areas of computer science in interesting ways that take advantage of the nature of the event annotation problem. Another contribution is the modified version of PLWAP (see section 3.6), which might have other applications. Other useful contributions include the modified way to measure similarity between words (section 3.3.2), dialogue lines (section 3.4), and sequences (section 3.9). The particular method of using affinity propagation clustering while updating similarities based on the current cluster assignments is also new.

Other, less tangible, contributions include some of the theories motivating the design of the system and its components, which were derived from research into related field, examination of human annotations of logs, and experimentation with different approaches. These theories help to formulate new concepts like "event-sequence," which might be a useful in other research. For example, the idea of expansion factor (section 3.6.1), pseudo-closed sequences (section 3.7) etc.

5.2 Future Work

The system is promising, but has a lot of room for improvement. This section describes many of the ideas I have had for how the system could be improved, but which I have not had the time to explore. The following subsections mention some of the other opportunities for future work on improving parts of the event-discovery system.

5.2.1 Large-Scale Modifications

Efficiency Improvements

The system in its current state is very slow, taking days to run from start to finish. The log pre-processing, cluster labelling, sequence mining, additional filtering and evaluation stages are all quite fast, taking at most a little over an hour combined. The word clustering, sequence-labelling and sequence clustering stages each take on the order of hours to a few 10s of hours to run. The dialogue and action clustering component takes the most time, typically about two days. This severely limits the amount of end-to-end testing and iteration that can be done and makes it difficult to evaluate the effect of changing earlier stages. The most time-consuming stages are the clustering and log labeling; candidate sequence mining is fairly fast due to all the filtering that takes place simultaneously. The clustering components spend the vast majority of their time computing and updating similarities.

Improving the speed of similarity computation would make the system much faster and allow for better evaluation and other possibilities. An easy way to greatly increase the speed would be to parallelize the process, which could be done easily since it is made up of millions of independent pair-wise similarity computations. Similarly the log labeling process could be easily parallelized. Porting some of the systems to 64-bit might also be useful because it would give them easier access to more memory, removing the current need to balance space with accuracy and efficiency in some cases (e.g. discarding similarities below a certain threshold, see section B.1). I did not spend much time optimizing my code in general, so it would be worth looking into in the future.

Parameter Optimization

One problem with the system is the massive number of parameters that it uses to control its different components (see Appendix B). I added these parameters over time based on research and to improve flaws I discovered in the system. The issue is that it is difficult to determine the optimal value for all these parameters. The system is very complex and the parameters interact in many ways, so it is not easy to evaluate the effect of changing a particular parameter. Another problem is that the only quantitative evaluation I have is for the system as a whole, which makes it even more difficult to set the parameters of an individual component.

In order to effectively experiment with different parameter values we would need to either improve the system's speed, as described in the previous section, or develop quantitative performance metrics for the output of the key components of the system. External cluster evaluation metrics could potentially be used with the clustering components to score them relative to gold-standard clusterings. The challenge there is creating gold-standard clusterings for tens of thousands of words or dialogue lines. One possibility might be to modify an external cluster evaluation metric to be able to use a gold-standard that is a subset of the total elements clustered. If changes in the parameters could be scored efficiently, then automatic parameter optimization might be possible. Genetic algorithms or other similar machine learning algorithms could be applied to the problem.

Human-Computer Collaboration

Although the ideal system would be able to completely automatically annotate the logs, the goal of the system is to reduce the amount of human effort needed as much as possible. To that end, it would be worthwhile to investigate the benefits of human involvement at different stages in the process. It is likely that a small amount of, possibly expert, human involvement early in the process would increase the accuracy of the system and greatly reduce the overall time spent by humans.

I did not have a chance to investigate this fully, but while working on the system I did notice many places where it seemed human-involvement would be useful. For example, during early experiments with the word and dialogue clustering systems, I spent about an hour revising the word clusters (e.g. combining separate clusters of numbers, geographical places, and names or splitting clusters that were too broad), and noted a decent improvement in the quality of the dialogue clustering. Also, while devising event-sequence filtering mechanisms, I noticed many kinds of sequences that a human could easily identify as non-event-sequences, but it was difficult to create a filtering rule to catch. If humans were to edit the list of event-sequences before labeling, it might save a lot of time.

The event-sequence annotation tool could also be modified to provide information back to the automatic system. For example, when someone labels a new event-sequence that the system did not catch, or modifies a sequence that the system labeled, the system could be notified and modify its labeling of all the other logs accordingly.

Larger Contexts and Smoothing

Due to limitations imposed by the space and time inefficiency of the system, the context similarity used in each of the clustering stages is not as effective as it could be. The contexts used are limited to at most two neighboring elements in the word case, and one in the dialogue and action case. If the system's efficiency were to be sufficiently improved or ported to 64-bit to allow for more memory to be used, it would be interesting to explore the effectiveness of larger contexts. It would also be worthwhile to explore smoothing possibilities as is typically done with n-grams, because currently there is no smoothing in the two-context-element word case.

5.2.2 Word Similarity and Clustering

Spell-Correction vs Surface Similarity

Context-based similarity measure have been shown to be effective for semantically clustering words in many studies (see section 2.4.1). In the event-sequence discovery system, surface similarity is also used when computing word similarities. The purpose was to catch and correctly cluster misspelling, typos and intentional abbreviations like "filet minion," "slamon," and "pls." In earlier experimental work with word clustering I has an initial spell-correction step, but discarded it because the system I was using was not the best and treated each word individually and so would miss many things like "minion." However, the current surface similarity solution sometimes clusters together words that shouldn't be grouped together, and misses others that should. For example, in table A.1, we can see that "salat" and "saland" are placed in the "salat" cluster rather than the "tart" cluster with the word "salad." Also, "ha" is placed in the same cluster as "he," when it should probably be with other laughing words like "lol" and "haha."

Adding an improved spell-correction step back in could have many potential benefits. First, it would greatly improve the speed of the word clustering stage because surface similarity would not have to be computed millions of times. Second, it would allow me to focus on the two problems of semantic clustering, and error corrections separately, probably improving the accuracy of both. Also, it would improve context similarity by having the incorrect words replaced with their correct versions prior to creating context features.

Expanded Dialogue Context

As noted previously in the word clustering analysis section (4.1), one class of words that my system does not cluster well is words that predominantly appear in one or two word utterances. This is because the majority of the word similarity measurement is based on context, and the in-utterance context of such words is uninformative. One way to fix this would be to expand the creation of contexts beyond single dialogue lines. This could potentially be done by using words from beginning and end of neighboring dialogue lines if the other lines were sufficiently close in time or some similar method.

5.2.3 Dialogue and Action Clustering

Word Clusters in Surface Similarity Computation

Although interesting in and of itself, the purpose of clustering words semantically is to assist with clustering dialogue lines semantically. This is implemented through dialogue line surface similarity computation where words of the same cluster are considered identical for the purpose of measuring the edit distance between two dialogue lines. However, this is the only way in which word clusters are used, if two words are not in the same

cluster than the IDF property of the word itself is used in computing edit distance, rather than properties of the cluster it belongs to.

Some modifications could allow dialogue surface similarity to make more use of the word clustering information. With some additional processing, IDF values could be computed for word clusters, and those could be used rather than individual word IDF values. A larger change could have the word clustering system output soft clusters, so words could belong to multiple clusters. This could give even more information about the semantic meaning of the word and the cost of a word swap could be weighted by the degree to which words share the same cluster(s). Another elaborate change could use clustering primarily to enhance the accuracy of similarity measurements between words and then swap costs could be computed from the similarity of the words directly rather than any cluster information. All of these changes would increase the time required to compute surface similarity, which is why I have not yet implemented them. Parallelization or other efficiency improvements would be required to implement significant improvements in dialogue surface similarity measurement.

Stop Words

In early iterations of the word clustering system I removed an automatically generated list of stop words from the dialogue lines prior to determining contexts. I later removed this due to the added complication of appropriately determining stop words and because the system was able to cluster stop words effectively anyway. However, it would be interesting to explore the possibility of removing stop words in order to simplify dialogue lines. This might make the surface similarity measurements between dialogue lines more indicative of their semantic meaning as stop words often vary or are omitted between two semantically equivalent phrases.

Semantic Analysis

So far I have avoided some of the typical semantic analysis techniques used in many sentence clustering applications (see section 2.4.2). This is because I did not believe they would be as effective as the techniques I used. They rely primarily on the information contained within the sentences themselves, and much of the dialogue I am working with consists of short utterances with little self-contained meaning. However, it would certainly be interesting to investigate the effectiveness of some of these techniques at least as an added component of overall similarity computation.

5.2.4 Candidate Event-Sequence Discovery

Filtering Improvements

Earlier I mentioned that I noticed many candidate event-sequences that the system produces as obviously invalid event-sequence (in section 5.2.1 under "Human-Computer Collaboration,"). I found it quite challenging to come

up with ways to filter out all these false positives. However, I have not spent enough time on this part of the system as I could have and there are probably a lot of ways the filtering system could be improved. Further analysis of the properties of the false positives that slip through that make them obviously false positives is in order.

Dynamic Bayesian Networks

In section 2.3.2, I discussed my early experiments with DBNs as a means to generate candidate event-sequences. I also explained that I abandoned them for the time being due to scalability issues. However, I still think they are very promising and would like to try to implement a custom DBN structure learning algorithm that takes advantage of the properties of my data to improve its efficiency.

Timestamps

Currently the event-sequence mining and filtering process treats the game logs as sequences of integers and searches them for patterns. These integers represent dialogue and action clusters, but there is other information in the original game logs that is not being used. Each action and dialogue line is tagged with a time-stamp, indicating the point in time when it occurred. Intuitively, actions and dialogue lines corresponding to the same event-sequence should in general have less time between them than unrelated actions and dialogue lines. This idea could be used during candidate sequence mining to filter sequences both in terms of a maximum average time between consecutive game-actions in the sequence, and a maximum temporal length of the sequence. I believe this idea is very promising and easy to implement, so it is probably the first thing I will try in future work. The game logs also contain information about the location where agents are at the time of particular dialogue lines or actions. This information could be used filter out sequences occurring in two different places simultaneously (e.g. the customer and waitress doing two separate things at the same time).

Object and Term Resolution

Another thing that candidate sequence mining currently ignores is the objects and entities involved in an event-sequence. This information is lost in the process of dialogue and action clustering. It is a very reasonable idea to believe that game-actions in a particular event-sequence will involve the same object or small set of objects. Reckman has has success in using the Restaurant Game data to associate terms with in game objects [27]. It would be very interesting to use her methods to figure out what actions and dialogue lines deal with the same objects. This could be used to filter out sequences dealing with too many separate objects or somehow provide a positive weight to object-coherent sequences, perhaps during labeling.

5.2.5 Event-Sequence Labeling and Clustering

Iterative Labeling

As described in section 3.8, part of the optimality criteria used to determine the best labeling of a log, is the normality of the sequences composing the labeling. Currently the normality is only computed in terms of the expansion factor of the sequence instances relative to the average for all instances of the same sequence. This is only because at this stage in the system there are no other useful features to base normality on. Most of the features that vary from instance to instance are gathered from the labeled logs, such as context features, and location in the log. It would be interesting to explore the possibility of iteratively labeling the logs. After each iteration, the average context and location features could be computed for each event-sequence and use to compute normality in the next labeling. Another variation would be to include sequence clustering in the iterative process as well and compute normality using the average features from the clusters the sequences belongs to.

Optimality Improvements

One component of the system that I did not spend as much time on as I would have liked is the labeling system. The optimality criteria used to select the best labeling of a log was primarily based off my intuition. I have not had the opportunity to carefully examine golden labellings and various automatic labellings well enough to refine how labellings are scored.

Inexact Sequence Matches During Labeling

Since the existing incarnation of the event-sequence discovery system only finds frequent event-sequences, many of the event-sequences in logs can not be correctly labeled. Although these unlabeled sequences are infrequent they are often quite similar to frequent sequences that can be labeled. Consequently, I have been contemplating possible ways we might be able to label these sequence based using their similarity to identifiable sequences. I have not fully thought it through, but one approach might be to use timestamps and object recognition to identify chains of likely related game-actions and compare them to the sequence database. If they are sufficiently similar to a sequence in the database they can be tentatively labeled with that sequence's event. Another possible approach might use subsequences of recognized sequences, testing to see if the subsequences can be grown to resemble the original sequences by using nearby related game-actions in the game log. Ultimately the idea is that valid event-sequence are too varied to be entirely caught by a frequent pattern mining approach.

Feature Similarity Improvements

Sequence similarity is composed of three parts, context similarity, surface similarity, and feature similarity. The latter compares vectors representing several features of the sequences, namely where in the game logs

they occur (beginning, middle or end) and their average expansion factor. There are probably other features of sequences that trend with the event the sequences correspond to. For example, the objects or actors involved in the game-actions of the sequence probably vary more between events than within them.

5.2.6 Evaluation Methods

Custom Extrinsic Cluster Measure

Finding a way to quantitatively evaluate the performance of the system was surprisingly a lot harder than I thought it would be. I think there is still a lot of room for improvement in the evaluation measures. I used extrinsic cluster evaluation metrics to evaluate performance based on the idea that elements in the same golden sequence should be in the same automatic sequence and treated it as a clustering problem. However, I am not completely convinced this is the best means to evaluate the labellings. Furthermore, I used existing extrinsic clustering metrics, which may not be ideal for the rather particular case I am working with. For example, I do not know what effects the fact that one class and cluster is much larger than the others (the other / not in a sequence label) has on the metrics. The small size of most clusters and the difference in numbers between clusters and classes may also have effects I am unaware of. It would be wise to investigate the extrinsic cluster measure in more detail to ensure they are measuring what I want, and perhaps modify or make a custom measure more appropriate for this situation.

Human Evaluation

Ultimately human evaluation would be the best way to evaluate the system because humans have a good intuition of what makes a good labeling, which is hard to capture in an algorithm. Humans could be presented with golden-automatic labeling pairs or excerpts and be asked to rank the quality of the automatic system on a scale. Or we could have humans correct the automatic labeling and rank them in terms of the number of changes made. It might be harder to evaluate the sequence clustering using humans. Also, this could not be easily used when fast evaluation is needed, such as for genetic algorithms on the parameters, but it would be a good way to evaluate the final system at some point.

Human evaluation would also be useful earlier in the system, not just to determine the overall performance. For example, along the lines of human-machine collaboration, humans could assist in labelling semantically meaningful clsuters for the various clustering stages. Or they could be used to cluster a subset of the elements, which could be compared to the automatic clustering using specialized extrinsic cluster measures.

5.3 Final Thoughts

Although the event-discovery system is not yet at a point where it could completely replace human annotation, the results are quite promising. The current state of the system does well enough at event-sequence labeling and clustering to significantly reduce the amount of human effort needed to completely annotate the dataset (see section 4.5). In addition, there is a lot of room for improvement in the system, it is conceivable that further development efforts could bring a new iteration of the system almost to the level of a human annotator. However, the sheer variation in human dialogue and behavior along with the amazing interpretation capabilities of the human mind make it unlikely that a computer system could fully replace human annotation work in the near future.

Appendix A

Sample Output

The following sections show some sample output from each of the key stages in the event-discovery system. The output has been modified from its original form to be more human-readable.

A.1 Sample Word Clusters

13		
11	14	23
12	15	starve
13	16	x
3		
1	50	0
10	6	few
2	7	five
3	8	three
4	80	two
5	9	
3rd		
1st	5th	high
2nd	elementary	sign
3rd	grade	third
another		
another	many	these
best	more	those
cash	new	virtual
different	other	
last	others	
ate		
ask	live	seem
asked	love	stole
ate	made	took
came	make	use
care	ordered	wanted
come	put	work
forgot	said	write
had	sat	
hate	see	
awesome		
alright	handsome	strange
awesome	hawt	sweet
awful	interesting	weird
cute	nasty	welcome
fantastic	perfect	yummy
fast	popular	
corner		
bil	floor	player
chair	kitchen	rocks
cheff	liking	servey
clouds	machine	sky
cops	management	summer
corner	manager	truth

corner contd.		
counter	microwave	window
delay	moon	wishes
dr	note	
flavour	owner	
customers		
chef	flowers	tables
customer	people	things
customers	players	waitresses
danke		
34	dave	noooooo
absolutely	definately	nty
ack	exactly	okaz
awsome	excelent	pancakes
berrypie	exelent	peace
brilliant	fag	roger
buttsecks	gladly	rum
byeeeee	gnite	salada
cali	goodday	sandwich
caroline	gotcha	shure
cheeseburger	hablo	slap
chuckles	hahaa	slurp
cofe	iamoo	spagehtti
coffeee	james	spagghetti
cries	jenny	splendid
dam	kay	tennessee
dang	kinky	tequila
dank	lobter	thankye
danke	maryland	thnaks
danku	michigan	wassup
darn	nee	
florida		
america	female	mo
canada	florida	tv
com	london	yo
germany		
belgium	england	maine
boston	france	san
cambridge	germany	spain
carolina	holland	texas
great		
bad	first	next
better	free	nice
delicious	good	pleasant
every	great	real

great contd.		
excellent	lovely	wonderful
fine	much	
hand		
arm	hands	pants
hand	mouth	
he		
ha	hes	they
he	she	who
hello		
bye	lol	sure
haha	no	thanks
hello	oh	thankyou
hey	ok	well
hi	okay	yeah
how	sorry	yes
kelly		
attend	german	kiki
bearguy	holla	maria
begin	horny	obviously
decision	janet	possibly
deposit	keisha	shellfish
dumbass	kelly	wellcome
may		
because	may	why
can	maybe	
let	shall	
minute		
forever	moment	second
minute	sec	
nectarine		
berry	nacterine	necterine
cherry	nectar	nectorine
cobb	nectarin	tangerine
grilled	nectarine	
nactarine	nectartine	
okey		
bien	non	spent
bon	ns	tres
bonjour	o	voila
fair	okey	
muy	oui	
portugal		
alberta	europe	italy
australia	greece	nj

portugal contd.		
brasil	hector	portugal
brazil	iceland	stranger
czech	ireland	
pretty		
ally	little	rather
extremely	most	really
grey	pretty	theyre
kinda	quite	
salat		
card	listening	taxi
cheak	meny	whatever
chips	pice	zitten
drag	saland	
feast	salat	
sixteen		
35	eighteen	rape
38	fifteen	several
45	hs	sixteen
couple	million	ten
soda		
beer	milk	tea
bourbon	pop	tonic
cocktail	shortly	vodka
coke	skeet	water
cunt	soda	whiskey
god	sour	whisky
tart		
cake	salmon	tarts
cheesecake	soup	pie
lobster	steak	
salad	tart	
white		
filet	red	white
house	whit	
wife		
bf	dad	husband
boyfriend	developer	immediately
bucket	dog	parents
crust	girlfriend	wife
your		
a	his	ur
any	my	your
big	some	youre
for	the	

Table A.1: Sample word clusters from the ouput corresponding to the parameters given in section B.1. The heading of each clsuter is the word that was chosen as the exemplar for that cluster by the algorithm. The clusters and words are arranged alphabetically top-to-bottom, left-to-right.

A.2 Sample Dialogue and Action Clusters

A.2.1 Sample Action Clusters

Pickup pot					
Actor	Action	Object	Precondition	Effect	Count
Waitress	Eat	Pot	Pot on ground		4
Customer	Give	Pot	Customer holding pot	Waitress holding pot	10
Waitress	Pickup	Pot	Pot on stove	Waitress holding pot	104
Waitress	Pickup	Pot	Pot on table	Waitress holding pot	6
Waitress	Pickup	Pot	Pot on floor	Waitress holding pot	4
Customer	Pickup	Vase	Vase on floor	Customer holding vase	5
Waitress	Pickup	Pan	Pan on table	Waitress holding pan	7
Waitress	Pickup	Pot	Pot on microwave	Waitress holding pot	2
Waitress	Pickup	Pot	Pot on counter	Waitress holding pot	8
Waitress	Pickup	Pot	Pot on dishwasher	Waitress holding pot	2
Waitress	Pickup	Pot	Pot on ground	Waitress holding pot	2
Waitress	Pickup	Pan	Pan on dishwasher	Waitress holding pan	1
Waitress	Pickup	DIRTY DISH	DIRTY DISH on pot	Waitress holding DIRTY DISH	2
Customer	Put down	Fruit	Customer holding fruit	Fruit on pan	1
Customer	Sit-on	Podium	Customer standing	Customer sitting on podium	27
Waitress	Sit-on	Register	Register on flowers	Waitress sitting on register	1
Waitress	Sit-on	Cuisinart	Cuisinart on fruit	Waitress sitting on cuisinart	1
Pickup DIRTY DISH					
Actor	Action	Object	Precondition	Effect	Count
Customer	Eat	DIRTY DISH	DIRTY DISH on blender, Customer sitting on CHAIR		2
Customer	Give	DIRTY DISH	Customer holding DIRTY DISH, sitting on chair	Waitress holding DIRTY DISH	3
Waitress	Look-at	DIRTY DISH	Waitress standing, DIRTY DISH on ground		2
Waitress	Pick-up	DIRTY DISH	Waitress standing, DIRTY DISH on table	Waitress holding DIRTY DISH	2320
Waitress	Pick-up	DIRTY DISH	Waitress standing, DIRTY DISH on floor	Waitress holding DIRTY DISH	68
Waitress	Pick-up	DIRTY DISH	Waitress standing, DIRTY DISH on OLD FOOD	Waitress holding DIRTY DISH	3
Waitress	Pick-up	Fruit bowl	Waitress sitting on chair, Fruit bowl on table	Waitress holding fruit bowl	5
Waitress	Pick-up	Bill	Waitress standing, Bill on OLD FOOD	Waitress holding bill	1
Waitress	Pick-up	DIRTY DISH	Waitress standing, DIRTY DISH on OLD FOOD	Waitress holding DIRTY DISH	7
Waitress	Pick-up	DIRTY DISH	Waitress standing, DIRTY DISH on podium	Waitress holding DIRTY DISH	1
Customer	Put-down	Bill	Customer sitting on DIRTY DISH holding bill	Bill on chair	1
Waitress eat OLD FOOD					
Actor	Action	Object	Precondition	Effect	Count
Customer	Give	OLD FOOD	Customer sitting holding OLD FOOD	Waitress holding OLD FOOD	1
Waitress	Eat	OLD FOOD	Waitress standing holding OLD FOOD	OLD FOOD is bitten	57
Customer	Eat	Register	Customer standing, waitress holding register		2
Customer	Get off	Trash	Customer sitting on trash	Customer standing	1
Customer	Look at	Bill	Customer standing, bill on trash		1
Waitress	Look at	Foodprep	Waitress standing, foodprep on counter		1
Customer	Pickup	Vase	Customer standing, vase on blender	Customer holding vase	1
Waitress	Pickup	Blender	Waitress standing, blender on flowers	Waitress holding blender	1
Customer	Pickup	Trash	Customer standing, trash on OLD FOOD	Customer holding trash	1

continued on next page ...

Actor	Action	Object	Precondition	Effect	Count
			Customer eat FOOD		
Actor	Action	Object	Precondition	Effect	Count
	Delete	Register	Customer holding register	Register deleted	7
Customer	Eat	BEVERAGE	Customer sitting, BEVERAGE on table	BEVERAGE sipped	3082
Customer	Eat	BEVERAGE	Customer sitting, BEVERAGE on table, sipped once	BEVERAGE sipped	2979
Customer	Eat	BEVERAGE	Customer sitting, BEVERAGE on table, sipped twice	BEVERAGE sipped	2685
Customer	Eat	OLD FOOD	Customer sitting, OLD FOOD on table	OLD FOOD bitten	767
Customer	Eat	OLD FOOD	Customer sitting, OLD FOOD on table, bitten once	OLD FOOD bitten	778
Customer	Eat	OLD FOOD	Customer sitting, OLD FOOD on table, bitten twice	OLD FOOD bitten	770
Customer	Eat	FOOD	Customer sitting, FOOD on table	FOOD bitten	3479
Customer	Eat	FOOD	Customer sitting, FOOD on table, bitten once	FOOD bitten	3463
Customer	Eat	FOOD	Customer sitting, FOOD on table, bitten twice	FOOD bitten	3406
Customer	Eat	BEVERAGE	Customer sitting holding BEVERAGE	BEVERAGE sipped	920
Customer	Eat	BEVERAGE	Customer sitting holding BEVERAGE, sipped once	BEVERAGE sipped	767
Customer	Eat	BEVERAGE	Customer sitting holding BEVERAGE, sipped twice	BEVERAGE sipped	645
Customer	Eat	FOOD	Customer standing, FOOD on table, bitten twice	FOOD bitten	92
Customer	Eat	FOOD	Customer standing, FOOD on chair	FOOD bitten	2
Customer	Eat	DIRTY DISH	Customer sitting, DIRTY DISH on table		1258
Customer	Eat	WINE	Customer sitting, WINE on table	WINE sipped	175
Customer	Eat	DIRTY DISH	Customer sitting holding DIRTY DISH		180
Customer	Eat	Flower	Customer sitting, flower on table	Flower bitten	893
Customer	Eat	Table	Customer sitting		371
Waitress	Eat	Flower	Waitress standing, flower on table	Flower eaten	340
Customer	Eat	Menu	Customer sitting, menu on table		150
Customer	Eat	Vase	Customer sitting, vase on table		213
Customer	Eat	WINE	Customer sitting, WINE on OLD FOOD	WINE sipped	3
Customer	Eat	Waitress	Customer sitting, Waitress standing		184
Waitress	Eat	DIRTY DISH	Waitress standing, DIRTY DISH on table		81
Waitress	Eat	BEVERAGE	Waitress standing, BEVERAGE on floor, sipped twice	BEVERAGE sipped	1
Waitress	Eat	FOOD	Waitress sitting, FOOD on ground, bitten twice	FOOD bitten	1
Customer	Eat	DIRTY DISH	Customer sitting on BEVERAGE holding DIRTY DISH		1
Customer	Eat	BEVERAGE	Customer sitting, BEVERAGE on floor, sipped twice	BEVERAGE sipped	1
Customer	Eat	WINE	Customer standing, WINE on blender, sipped twice	WINE sipped	1
Customer	Eat	FOOD	Customer sitting on bartender holding FOOD, FOOD bitten once	FOOD bitten	4
Customer	Eat	FOOD	Customer sitting on bartender holding FOOD, FOOD bitten twice	FOOD bitten	4
Waitress	Eat	FOOD	Waitress sitting on FOOD, FOOD on counter	FOOD bitten	2
Customer	Eat	Trash	Customer sitting		1
Waitress	Eat	FOOD	Waitress standing, FOOD on OLD FOOD, bitten once	FOOD bitten	1
Customer	Eat	Menu	Customer sitting, menu on fruit		1
Customer	Eat	Waitress	Customer sitting on counter, waitress sitting on FOOD		1

continued on next page ...

Actor	Action	Object	Precondition	Effect	Count
Waitress	Get off	Register	Waitress sitting on register on table	Waitress standing	1
Customer	Get off	Flower	Customer sitting on flower on podium	Customer standing	1
Customer	Look at	FOOD	Customer sitting, FOOD on table		160
Customer	Look at	BEVERAGE	Customer sitting, BEVERAGE on table		86
Customer	Look at	Flower	Customer sitting, flower on table		253
Customer	Look at	DIRTY DISH	Customer sitting, DIRTY DISH on table		76
Customer	Pay	Bill	Customer standing, bill on OLD FOOD		1
Customer	Pickup	Flower	Customer sitting, flower on table	Customer holding flower	476
Customer	Pickup	FOOD	Customer sitting, FOOD on table	Customer holding FOOD	652
Customer	Pickup	BEVERAGE	Customer sitting, BEVERAGE on table, sipped once	Customer holding BEVERAGE	317
Customer	Pickup	BEVERAGE	Customer sitting, BEVERAGE on table, sipped twice	Customer holding BEVERAGE	187
Waitress	Pickup	Trash	Waitress sitting on fridge, trash on fridge	Waitress holding trash	1
Customer	Pickup	WINE	Customer standing, WINE outside	Customer holding WINE	1
Waitress	Pickup	FOOD	Waitress standing, FOOD on floor, bitten twice	Waitress holding FOOD	6
Customer	Pickup	BEVERAGE	Customer sitting, BEVERAGE on floor, sipped twice	Customer holding BEVERAGE	1
Waitress	Pickup	Trash	Waitress standing, trash on food	Waitress holding trash	1
Customer	Pickup	Trash	Customer sitting, waitress holding trash	Customer holding trash	1
Customer	Put down	FOOD	Customer sitting holding FOOD, bitten once	FOOD on table	107
Customer	Put down	BEVERAGE	Customer sitting holding BEVERAGE	BEVERAGE on table	518
Customer	Put down	BEVERAGE	Customer sitting holding BEVERAGE, sipped once	BEVERAGE on table	451
Customer	Put down	BEVERAGE	Customer sitting holding BEVERAGE, sipped twice	BEVERAGE on table	286
Customer	Put down	DIRTY DISH	Customer sitting holding DIRTY DISH	DIRTY DISH on table	835
Customer	Put down	FOOD	Customer sitting holding FOOD	FOOD on table	890
Customer	Put down	Flower	Customer sitting holding flower	Flower on table	250
Customer	Put down	Fruit	Customer standing holding fruit	Fruit on trash	1
Customer	Put down	WINE	Customer standing holding WINE	WINE on bar	4
Customer	Touch	BEVERAGE	Customer sitting, BEVERAGE on table		129
Customer	Touch	Waitress	Customer standing, waitress standing		353
Customer	Touch	Flower	Customer sitting, flower on table		54
Customer	Touch	Table	Customer sitting		57

Waitress clean DIRTY DISH

Actor	Action	Object	Precondition	Effect	Count
Waitress	Clean	DIRTY DISH	DIRTY DISH on counter	DIRTY DISH cleaned	3373
Waitress	Clean	FOOD	FOOD on counter	FOOD cleaned	184
	Delete	DIRTY DISH	DIRTY DISH on bar	DIRTY DISH deleted	1513
	Delete	DIRTY DISH	DIRTY DISH on counter	DIRTY DISH deleted	3621
	Delete	BEVERAGE	BEVERAGE on counter	BEVERAGE deleted	327
	Delete	BEVERAGE	BEVERAGE on bar, one sip taken	BEVERAGE deleted	219
	Delete	BEVERAGE	BEVERAGE on bar, two sips taken	BEVERAGE deleted	159
	Delete	FOOD	FOOD on counter, two bites taken	FOOD deleted	192
Waitress	Eat	DIRTY DISH	Waitress standing, DIRTY DISH on floor		1
Customer	Get off	Trash	Customer sitting on trash on foodprep	Customer standing	1
Waitress	Get off	Fruit bowl	Waitress sitting on fruit bowl	Waitress standing	1
Customer	Give	DIRTY DISH	Customer sitting holding DIRTY DISH	Waitress holding DIRTY DISH	197
Customer	Give	DIRTY DISH	Customer sitting on BEVERAGE holding DIRTY DISH	Waitress holding DIRTY DISH	3
Waitress	Pickup	DIRTY DISH	Waitress standing, DIRTY DISH on table	Waitress holding DIRTY DISH	3232
Waitress	Pickup	FOOD	Waitress standing, FOOD on table	Waitress holding FOOD	361
Customer	Pickup	DIRTY DISH	Customer sitting, DIRTY DISH on table	Customer holding DIRTY DISH	812

continued on next page ...

Actor	Action	Object	Precondition	Effect	Count
Waitress	Pickup	FOOD	Waitress standing, FOOD on table, bitten twice	Waitress holding FOOD	177
Waitress	Pickup	DIRTY DISH	Waitress standing, DIRTY DISH on microwave	Waitress holding DIRTY DISH	4
Waitress	Pickup	Trash	Waitress standing, trash on OLD FOOD	Waitress holding trash	1
Waitress	Pickup	Fruit	Waitress standing, fruit on trash	Waitress holding fruit	2
Customer	Pickup	Microwave	Customer sitting on trash, microwave on floor	Customer holding microwave	1
Waitress	Pickup	Menu	Waitress standing, menu on counter	Waitress holding menu	167
Waitress	Pickup	Fruit bowl	Waitress standing, customer holding fruit bowl	Waitress holding fruit bowl	5
Waitress	Pickup	Table	Waitress standing		657
Customer	Pickup	Table	Customer sitting		138
Waitress	Put down	DIRTY DISH	Waitress standing holding DIRTY DISH	DIRTY DISH on counter	3273
Waitress	Put down	DIRTY DISH	Waitress standing holding DIRTY DISH	DIRTY DISH on bar	1078
Waitress	Put down	DIRTY DISH	Waitress standing holding DIRTY DISH	DIRTY DISH on table	471
Waitress	Put down	DIRTY DISH	Waitress standing holding DIRTY DISH	DIRTY DISH on floor	245
Waitress	Put down	Vase	Waitress standing holding vase	Vase on bartender	4
Customer	Put down	EMPTY WINE	Customer sitting holding EMPTY WINE	EMPTY WINE on flower	1
Customer	Put down	Fruit	Customer standing holding fruit	Fruit on dishwasher	1
Customer	Sit on	Fruit bowl	Customer standing, fruit bowl on counter	Customer sitting on fruit bowl	2
Customer	Sit on	Flower	Customer standing, flower on fridge	Customer sitting on flower	1
Customer	Touch	DIRTY DISH	Customer sitting, DIRTY DISH on table		72
Waitress	Touch	FOOD	Waitress standing, FOOD on table		2
Customer	Touch	WINE	Customer standing, waitress holding WINE		2

Waitress pickup FOOD

Actor	Action	Object	Precondition	Effect	Count
	Delete	Bill	Bill on register	Bill deleted	3115
	Delete	Menu	Menu on podium	Menu deleted	219
Customer	Fail pay	Bill	Customer standing, bill on OLD FOOD		1
Waitress	Get off	CHAIR	Waitress sitting on CHAIR	Waitress standing	1077
Customer	Get off	Register	Customer sitting on register on counter	Customer standing	2
Waitress	Look at	Menu box	Waitress standing		333
Waitress	Look at	Menu	Waitress standing, menu on counter		56
Waitress	Look at	Bill	Waitress standing, bill on OLD FOOD		1
Waitress	Pickup	FOOD	Waitress standing, FOOD on counter	Waitress holding FOOD	3827
Waitress	Pickup	BEVERAGE	Waitress standing, BEVERAGE on bar	Waitress holding BEVERAGE	3928
Waitress	Pickup	Flower	Waitress standing, flower on fruit	Waitress holding flower	2
Waitress	Pickup	OLD FOOD	Waitress standing, OLD FOOD on OLD FOOD	Waitress holding OLD FOOD	2
Customer	Pickup	Fruit bowl	Customer sitting, fruit bowl on menu	Customer holding fruit bowl	1
Customer	Pickup	EMPTY WINE	Customer sitting, EMPTY WINE on OLD FOOD	Customer holding EMPTY WINE	1
Waitress	Put down	FOOD	Waitress standing holding FOOD	FOOD on table	3469
Waitress	Put down	BEVERAGE	Waitress standing holding BEVERAGE	BEVERAGE on table	3379
Waitress	Put down	WINE	Waitress standing holding WINE	WINE on table	258
Waitress	Put down	Menu	Waitress standing holding menu	Menu on counter	418
Waitress	Put down	Flower	Waitress standing holding flower	Flower on table	644
Customer	Put down	Vase	Customer sitting holding vase	Vase on table	109
Waitress	Put down	Vase	Waitress standing holding vase	Vase on table	235
Customer	Put down	BEVERAGE	Customer standing holding BEVERAGE	BEVERAGE on register	1
Customer	Put down	Menu	Customer sitting holding menu	Menu on fruit	1
Customer	Sit on	Table	Customer standing	Customer sitting on table	263
Customer	Touch	FOOD	Customer sitting, FOOD on table		51
Waitress	Touch	Customer	Waitress standing, customer sitting on BEVERAGE		1

continued on next page ...

Actor	Action	Object	Precondition	Effect	Count
Customer	Touch	FOOD	Customer sitting on table, FOOD on CHAIR		1
			Pay bill		
Actor	Action	Object	Precondition	Effect	Count
Waitress	Eat	Bill	Bill on table		2
Customer	Eat	Bill	Customer sitting, Bill on table		2
Customer	Eat	Bill	Customer sitting		3
Customer	Fail-pay	Bill	Customer sitting		79
Customer	Give	Bill	Customer holding bill	Waitress holding bill	458
Customer	Look at	Bill	Customer sitting, holding bill		102
Customer	Look at	Bill	Customer standing, bill on table		17
Waitress	Look at	Bill	Waitress standing, bill on table		116
Waitress	Look at	Bill	Waitress standing, customer holding bill		96
Customer	Look at	Bill	Customer sitting, bill on table		53
Customer	Pay	Bill	Customer sitting	Bill paid	1327
Customer	Pay	Bill	Customer sitting, bill on table	Bill paid	548
Customer	Pay	Bill	Customer sitting, bill on table, bill paid	Bill paid	1
Waitress	Pickup	Bill	Waitress standing, bill on table, bill paid	Waitress holding bill	921
Waitress	Pickup	Bill	Waitress standing, bill paid	Waitress holding bill	811
Customer	Pickup	Bill	Customer sitting, bill on table, bill paid	Customer holding bill	97
Customer	Pickup	Bill	Customer sitting, waitress holding bill, bill paid	Customer holding bill	5
Waitress	Pickup	Register	Waitress standing	Waitress holding register	41
Customer	Pickup	Vase	Customer sitting, vase on flowers	Customer holding vase	2
Customer	Pickup	Fruit	Customer sitting, fruit on table	Customer holding fruit	5
Customer	Pickup	Bill	Customer sitting, bill on chair	Customer holding bill	7
Waitress	Pickup	WINE	Waitress standing, WINE on flower	Waitress holding WINE	1
Customer	Put down	Bill	Customer standing	Bill on table	92
Waitress	Put down	Fruit bowl	Waitress standing	Fruit bowl on OLD FOOD	1

A.2.2 Sample Identical Dialogue Lines

The dialogue clustering is performed in two stages. First dialogue lines that are identical in terms of the word cluster sequence they are composed of are grouped together. Then the dialogue lines are clustered using context and surface similarity along with the actions. The following are representative sample identical line pairs from the first stage. [1]

Table A.2: Sample action clusters output by the dialogue and action system using the parameters given in table B.4. The capitalized words and phrases represent clusters of objects that were generated in the log processing step (see section 3.2). The counts are the number of occurrences in 5000 game logs.

can i get a beer ?	can i have some water ?
how can i help you ?	how may i serve you ?
coffie please	beer please
coffe please	beer please
some water please	a beer please
spaghetti please	steak please
cheese pie please	filet minon please
white wine	red wine
comin right up	coming right up
may i get you something to drink ?	can i get u anything to drink ?
right up	right away
here is your beer	here is the wine
house	red
did you need something ?	do you need anything ?
how is everything ?	how was everything ?
spahgetti marinara	spaghetti marinara

continued on next page ...

[1] Note that all the sample dialogue lines presented in this thesis are in their post-processed from. Meaning they are in lower-case and have all punctuation except "?" removed

there u go	there ya go
okay fine	ok good
whats your age ?	whats ur age ?
well ?	so ?
what do you need	what do you want
a lobster	your filet
something to drink	anything to drink
ok mister	yes sir
would you like some more beer ?	would you like some more wine ?
hows everything going ?	hows it going ?
sit	help
how bout a wine	how bout a beer
how do i work this ?	how do you work it ?
would you like a table	do you want a table
can i take the menu ?	shall i take the menu ?
the lobster will be right here	the lobster will be right up
just a moment please	just a sec pls
white whine	red wine
would you like the check now	do you want the check now
one second please	one moment please
steak and cheesecake	salad and lobster
your cheesecake sir	your lobster sir
would you like anything else	would you like something eles
here is todays menu	here is our menu
welcome in our restaurant	welcome to our restaurant
please take a sit	please take your sit
any food	your dinner
can i get you wine or beer ?	can i get you tea or coffee ?
youre most welcome	your most welcome
yup sorry	ok sorry
wisconsin u ?	uk you ?
would you liek to order something ?	would you like to order something ?
thank you for stopping by have a good evening	thank you for stopping by have a great day
more water ?	another beer ?
ahh i see	ohh i see
enjoy your filet sir	enjoy your pie sir
heres your bill sir	heres the bill sir

Table A.3: Sample identical waitress dialogue line pairs. Each row represents a pair of dialogue lines uttered by the waitress found to be identical in terms of word clusters.

waitress	woman
perfect	mmm
oh thank you	ah thank you
hmm	hmmm
yummy	mmm
it was very good	this is quite nice
more beer please	another beer please
now ill take some lobster	now ill take some salmon
now ill take some steak	now ill take some salmon
oh ok	ah ok
now some water	now some wine
with spagghetti	and spaghetti
ohh thank you	ah thank you
i would like a berry pie	i would like a filet mignon
thaks	thx
no that will be all	no this should be all
ill have the cherry cheesecake and a coffee please	id like a cobb salad and a tea please
thats nice	looks good
sure cheesecake	yes salad
fine thanks	mm thanks
have a good day	have a nice day

continued on next page ...

yes wine please	ok wine please
thanks sweetie	thanks babe
can i get some wine please	can i have a beer please
can i give my order	can i tell the order
everything was great	everything is fine
can i get the check please	may i have the bill please
thanks for the great hospitality	thanks for the fine service
that sounds great	this looks great
that looks great	this looks great
no thank you just the check	no thank you just a check
brasil	germany
no just the bill	no just the check
thak you very much	thank you very much
table for 1	table for one
a filet mignon please	a cobb salad please
i want a spaghetti	i need a lobster
it was lovely	everything is fine
could i have the bill please	could i get the check please
the salmon was wonderful	the salad was good
could i have the check ?	could i have the bill ?
hey cutie	hello babe
a menu please	the menu pls
no just the bill please	no just the check please
may i sit here ?	can i sit here ?
i would like the cobb salad and a coffee pleaes	can i have a filet mignon and a tea please
i would like the cobb salad and coffee please	id like a cobb salad and a tea please
its okay	im ok
lobster 2	cheesecake 2
1 salmon	one soup
1 salad	one soup
1 more lobster	1 more salad
where u from	where you from
wher are u from	where are you from
wassup	peace
whoa	wow
a beer would be great	a soda would be great
everything is just fabulous	this is just fine
yes i would like a soup	yes i would like the spaghetti
i think i ill have the lobster	i think i would like some cheesecake
that sounds good	this looks great
nice day	good evening
ill have the cheesecake	ill have the tart
looks delicious	looks good

A.2.3 Sample Dialogue Line Clusters

Sample Customer Dialogue Line Clusters

These are whole and partial sample dialogue line clusters. The bolded dialogue line is the exemplar for the cluster. Note that the dialogue lines listed may represent a set of multiple identical dialogue lines as described in the previous section.

- **thank you**
  - thank you please
  - thank you yet again
  - thank you dear
  - thank you beautiful

- thank you p
- thank you misses
- thanks but
- thanks rachel
- thanks alot
- thanks toots
- thanks kindly
- thanks slave
- thanks for that
- thanks for asking
- thanks a lot
- sorry for the trouble
- thanx a lot
- please and thank you mam
- thank you much
- thank you though
- thank you mam
- thank you darlin
- thank you babe
- thank you darling
- thank you kindly
- thank you slave
- thank you alice
- thank you jenni
- oh thank you mam
- why thank you maam
- hello and thank you
- thanks for a nice meal
- thanks for the fine service
- thanks have a nice day
- thanks have a good one
- how you like your tip ?
- bye and have a nice day
- keep up the good work
- thank you that looks very nice
- thank you food was wonderful
- thank you so much
- thank you very kind
- thank you very much
- thank you very much rachel
- thank you very much mam
- thank you very much beautiful
- thank you verry much
- thank you young man
- no thank you have a nice evening
- no im fine thanks
- okay thank you very much
- why thank you very much
- ohhh thank you very much

- thanks looks great
- thanks sooooo much
- i thank you
- thank you
- thank yo
- thank yo u
- thank zou
- thank yoiu
- thank yu
- ...

- **could i have the bill please ?**
  - i will take the check
  - the bill
  - or a blowjob
  - no just the check
  - no thanks just the bill
  - just the check
  - just the check please
  - just the check maam
  - then the bill
  - im ready for my bill please
  - ill have the bill please
  - ill have my check now please
  - ill just have the check please
  - id like my bill now
  - im ready for the check now
  - ill eat it anyway
  - may i have the check now
  - may i have my check now ?
  - may i have my check now natalie ?
  - let me get the check for you
  - gave you a tip too
  - the bill please
  - the bill now
  - the count please
  - and the bill please
  - check please
  - ty bill please
  - maam the bill please
  - umm check please
  - worst waitress ever
  - i have money
  - could i have the bill ?
  - could i have the bill please ?
  - k may i have my check please ?
  - can i get my check now ? ?
  - chef
  - ill pay my bill now
  - ill take the check please

  - ill take the check too please
  - ill take my check now
  - ill take the check
  - ill just take the check
  - ill just take the check please
  - no thank just the check
  - then i will take the bill
  - hold me now
  - i am ready for the bill
  - i think im ready for a check
  - i guess ill take the check
  - im ready for my bill
  - im out of money
  - please may i have the bill
  - i will not pay that bill
  - did you pay your bill
  - could i help you
  - lets see the bill
  - give me another bill
  - ...
- **wine please**
  - and spaghetti
  - to start
  - one more beer please
  - and more water please
  - yeah another beer would be nice
  - and wine
  - and chease cake please
  - and some coffe
  - and as a desert
  - and wheres my food
  - and berry pie for desert
  - a lobster and a cheesecake
  - a coffee
  - water and vegetable soup
  - cobb salad and filet mignon
  - cobb salad and grilled salmon
  - cobb salad spaghetti berry pie
  - vegtable soup and cobb salad
  - i will have a red wine
  - i will have a red wine please
  - i will have the filet
  - i would like a white wine
  - i just got fired from my job
  - i said white wine
  - beer lots of it
  - a steak and salad please
  - a lobster and a salad please
  - a cob salad please

- fillet mignon and cobb salad please
- and a coffee perhaps
- and more coffee
- and coffee please
- medium rare please
- goodbye honey
- pepper and salt please
- salty
- salad and water please
- soup lobster and tea please
- soup salad and lobster please
- soup de jour and a coffee
- soup de jour and filet mignon
- soup du jour and water
- soup du jour and water please
- vegetable soup and a pint of beer please
- i would like a beer and a salad please
- i would like a glass of water
- i would like a filet mignon please
- i would like a berry pie
- how about a beer
- how about a salad
- how about a coffee
- how about some tea ?
- hey get me a lobster
- ok now the steak
- could i have a berry pie and a glass of water ?
- yes a white wine please
- and some red wine please
- and a red wine
- and more wine
- and red wine
- and filet migon
- wine red wine
- the red wine
- sure red wine
- water and red wine
- white and red wine
- two glasses of red wine
- 2 red wine
- thank you for sitting on me
- can i have a beer with that ?
- can i have a soup du jour
- can i have the filet mignon ?
- can i have my check pleae ?
- can i get a nectarine tart please ?
- can i try the grilled salmon ?
- ...

- **im ready to order**

  - i am ready to order
  - i dont drink
  - i think im ready to order
  - i think im ready to order now
  - id like to have a salad
  - im ready to order
  - im ready to order now
  - im ready now
  - im from brasil
  - ok im ready to order

- **how old are you**
  - well how old r u
  - so how old r ya ?
  - how old r u ?
  - how old ru ?
  - 20 and you ?
  - chris
  - denise urs ?
  - nice 2 meet u
  - oh well im done
  - oh its alright
  - no im not
  - so how old r u
  - how about you
  - how old are you
  - how old are you how old are you
  - hey you touched me
  - shucks
  - same

- **just asking**
  - just curious
  - just asking
  - only joking
  - wont work
  - stay
  - kiss
  - hellooooooo
  - thankies
  - auf wiedersehen
  - wasser

Sample Waitress Dailogue Line Clusters

- **hello sir**
  - please do come in
  - hello sir and welcome table for one today ?
  - welcome to house of pies
  - hiii
  - hello come in
  - please come with me
  - welcome to good burger home of the good burger
  - welcome to goodburger home of the goodburger
  - excuse me sir ?
  - hi how you doing ?
  - so how are you today ?
  - how are you ?
  - how are you this fine morning ?
  - how are you this afternoon ?
  - how are you today sir ?
  - how are you today ?
  - how are you doing ?
  - how are you doing today sir ?
  - how are you doing tonight ?
  - how r u ?
  - how many are you ?
  - hey how are you ?
  - would you like a table for one sir ?
  - good day then
  - good evening sir
  - good evening and welcome
  - good choice senor
  - good afternoon sir
  - good afternoon madam
  - good eveing sir
  - come on sir
  - come on in
  - come in
  - come in please
  - come in follow me
  - come right in
  - welcome please come in
  - welcome to the restaurant
  - welcome to the restaurant sir
  - welcome to the cafe
  - welcome to our fine establishment
  - welcome to our restaurant
  - welcome to senor fancypants
  - welcome to ninja joes eatery
  - welcome to therestaurant

- welcome in this restaurant
- welcome how are you
- um are you ok
- hello sir welcome
- hello and welcome
- hello welcome
- oh hello welcome
- oh ahaha
- oh nvm
- hi sexy
- ...

- **would you like some time beforee you order ?**
  - whenever you are ready sir
  - why did you bite me ?
  - have you had a chance to look at the menu ?
  - just call me when you are ready
  - just call me when youre ready
  - just call if you need anything
  - when you are ready to order
  - tell me when you are ready
  - call me if you need anything
  - i thought i was a stalker ?
  - have u decided yet ?
  - have you decided ?
  - have you decided yet sir ?
  - have you already choosen ?
  - have you finished sir ?
  - have you choose ?
  - have you choosen ?
  - please tell me what you exactly want sir
  - please tell me when youre ready to order
  - ill give you a few moments
  - ill give you a minute to decide
  - do you want any thing to drink ?
  - just tell me when your ready
  - just tell me when youre ready to order
  - give me a holler when youre ready to order
  - let me know when youre ready to order
  - tell me when u r ready to order
  - tell me when your ready
  - komt eraan
  - are you ready to order or do you need a few minutes ?
  - are you ready to order or do you need some more time ?
  - are you ready to order or do you need more time ?
  - are you ready to order or need more time ?
  - just let me know when youre ready
  - just call me if you need anything
  - let me know if you need anything
  - let me know when you are ready

- let me know when you are ready for that
- let me know when your ready
- let me know wen you are ready to order
- have you decided on any food ?
- have you decided on anything ?
- u want something to eat ?
- you ready to order ?
- are you alright sir ?
- are you all done ?
- are you ready to order sir
- are you ready to order sir ?
- are you ready to order ?
- are you ready 2 order sir
- are you finished with that ?
- here you go are you ready to order ?
- no problem are you ready to order ?
- whenever you are ready
- whenever youre ready
- whenever youre ready sir
- ...

- **ok coming right up**
  - i will be right back with the wine
  - i will be right back with your beer
  - i will be right out with it sir
  - your lobster should be here shortly
  - your soup will be ready shortly
  - your filet will be right up
  - i will get it for you
  - i have that right up
  - ok ill get that for u
  - i will be back with your beer
  - i will be right back with your salad
  - will that be all for today
  - be right back with your salad
  - ill be right over here if you need anything
  - sure be right back with your soup
  - sure ill be right back with the wine
  - ok i will get that
  - ok ill just grab my food
  - ok ill go get it
  - ok ill fix it
  - ok let me get it
  - great choice one minute please
  - absolutely one moment please
  - ill have that right out to you
  - ill have it right over
  - ill get it for you now
  - ill bring that right out
  - ok i will have that right out

  - wonderful ill bring it right out
  - right away sir
  - right away kind sir
  - right away maam
  - right over here sir
  - come right away sir
  - coming up sir
  - coming right up sir
  - here is your soup ill be right back with your beer
  - all right ill be back in a moment
  - sure be right back with your beer
  - alright ill be right back with that
  - ill be right back to take your order
  - sure i will be right back with that
  - sure be right back with that
  - sure ill get that right away
  - sure ill bring it right to you
  - of course i will be back shortly
  - it will be right up
  - ok i will be right with you
  - ok i will be right back
  - right away
  - ok right away
  - great coming right up
  - coming up
  - coming right up
  - coming right up dude
  - coming rite up
  - coming righ up
  - all right just a moment
  - sure i will be a moment
  - sure i will give you a minute
  - sure itll just be a minute
  - ok just one minute sir
  - ok just a moment
  - ok just a moment sir
  - ok wont be long
  - ok itll be just a moment
  - ...
- **this is on the house**
  - one beer on the house
  - compliments of the chef
  - compliments of the house
  - beer on the house
  - on the house
  - on the house sir
  - no problem take your time
  - water is on the house
  - happy birthday to you

  - refills on the house
  - get out of my shop
  - im taller then you
  - this one is on the house
  - this is a robbery
  - this is on the house
  - this ones on the house
  - its on the house
  - happy birthday
  - heres a lobster on the house
  - wines on the house
  - have some pie on the house
  - you can pay at the register
  - the salad is on the house
  - the pie is on the house
  - the bottles are on the house
  - this beer is on the house
- **my name is amanda i will be your waitress today**
  - i am sarah
  - the table is good
  - my name is sally
  - my name is shayla
  - my name is janet
  - my name is paula i will
  - my name is johanna
  - your dinner is ready
  - lovely evening isnt it
  - schnell
  - evan and yours ?
  - i will be your waitress today
  - my name is virginia and ill be your server today
  - my name is amanda i will be your waitress today
  - my name is amanda i will be serving you today
  - my name is sean i will be your server tonight
  - my name is miranda ill be your waitress today
  - my name is allison ill be serving you tonight
  - my name is janet and ill be your waitress for this evening
  - my name is paula and what is your name ?
  - im alice i will be your waitress today
- **look a floating plate**
  - look a floating plate
  - garbage
  - woo money
  - text follow me
  - menus
  - soft drink
  - common
  - hannah
  - purrrrrr
  - hy
  - agreed

A.3 Sample Candidate Sequences and Clusters

The following are sample clusters and partial clusters of candidate event-sequence that were mined from the game log. The sequences are given in terms of the exemplars of the clusters of actions and dialogue lines that make up the sequence. The clusters are labelled with a relevant event label.

- **Waitress serves customer food**
  - Waitress puts down food on table - Waitress says "here is your meal"
  - Waitress puts down food on table - Waitress says "here is your meal" - Customer picks up food
  - Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table
  - Waitress puts down food on table - Waitress says "here is your meal" - Customer eats food - Customer eats food - Customer eats food
  - Waitress puts down food on table - Customer says "thank you" - Customer eats food - Customer eats food - Customer eats food
  - Customer says "where is my salad" - Waitress says "ok coming right up" - Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table
  - Waitress puts down food on table - Waitress says "this is on the house" - Customer eats food - Customer eats food
  - Waitress gives blender to customer - Customer eats flowers
  - Waitress puts down food on table - Customer says "this looks good" - Customer eats food
  - Waitress looks at fridge - Customer eats trash
  - Waitress puts down food on table - Waitress says "that is filet mingon" - Customer eats food
  - Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table - Waitress says "here is your meal"
  - Waitress puts down food on table - Waitress says "anything else sir"
  - Waitress puts down food on table - Waitress says "anything else sir" - Customer looks at menu
  - Waitress says "beer" - Waitress picks up food from counter - Waitress puts down food on table - Customer eats
  - Waitress puts down food on table - Customer picks up food - Customer eats food - Customer eats register
  - Waitress picks up food from counter - Waitress puts down food on table - Waitress says "enjoy sir"
  - Waitress puts down food on table - Waitress says "enjoy sir" - Customer picks up food - Customer eats food - Customer eats food - Customer eats food
  - Waitress puts down food on table - Waitress says "enjoy sir" - Customer eats food
  - Waitress gives food to customer - Customer puts down food on table
  - Waitress puts down food on table - Customer says "thank you" - Customer picks up food - Customer eats food - Customer eats food
  - Waitress picks up food from counter - Waitress puts down food on table
  - ...
- **Waitress gives customer bill**
  - Waitress picks up dirty dish from table - Waitress says "I'll get your bill" - Waitress touches register - Waitress picks up bill - Waitress puts bill on table
  - Waitress picks up dirty dish from table - Waitress says "would you like your check"
  - Waitress touches register - Waitress says "I'll get your bill" - Waitress picks up bill
  - Customer says "could i have the bill please" - Waitress touches register - Waitress picks up bill - Waitress gives bill to customer
  - Customer says "could i have the bill please" - Waitress touches register - Waitress picks up bill - Waitress puts down bill on table - Customer looks at bill
  - Waitress puts down bill on table - Waitress picks up dirty dish
  - Customer says "could i have the bill please" - Waitress touches register - Waitress picks up bill - Waitress puts down bill on table - Customer picks up bill
  - Waitress puts down bill on table - Customer says "thank you"
  - Waitress touches register - Waitress picks up bill - Waitress touches customer - Customer touches waitress
  - Waitress picks up bill - Waitress says "here is your bill" - Waitress puts down bill - Customer looks at bill
  - Waitress puts down bill - Waitress says "the wine was on the house"

- Waitress touches register - Customer says "do I have to pay" - Waitress picks up bill - Waitress puts down bill on table
- Waitress touches register - Waitress picks up bill
- ...

- **Customer orders food**
  - Waitress puts down menu on table - Waitress says "the menu sir" - Customer picks up menu from table - Customer looks at menu
  - Customer looks at menu - Customer picks up Cuisinart
  - Waitress says "what would you like" - Customer says "spaghetti marinara" - Waitress picks up menu from table - Waitress puts down menu on menu box - Delete menu
  - Customer looks at menu - Customer says "spaghetti marinara" - Waitress says "ok coming right up"
  - Customer looks at menu - Customer puts down menu - Customer says "spaghetti marinara"
  - Customer puts down menu - Customer says "steak me" - Waitress picks up menu from table - Waitress puts down menu on menu box
  - Waitress says "what would you like" - Customer says "spaghetti marinara"
  - Customer looks at menu - Waitress says "what would you like" - Customer says "spaghetti marinara" - Customer gives menu to waitress - Waitress puts down menu on menu box - Delete menu
  - Customer picks up menu from table - Waitress says "what would you like" - Customer says "spaghetti marinara" - Customer looks at menu - Customer puts down menu on table - Waitress picks up menu - Waitress puts down menu on menu box
  - ...

- **Waitress greets customer**
  - Waitress says "hello sir" - Customer says "hello" - Waitress says "follow me"
  - Waitress picks up menu - Waitress says "hello sir"
  - Waitress says "hello" - Customer says "table for one please" - Waitress says "follow me" - Customer sits down
  - Waitress says "hello sir" - Customer says "hello" - Waitress picks up menu
  - Customer says "hello" - Waitress says "table for one"
  - Customer says "whats up" - Waitress picks up menu
  - ...

- **Waitress cleans table**
  - Waitress picks up dirty dish from table - Waitress puts down dirty dish on counter - Waitress cleans dirty dish - Delete dirty dish
  - Waitress says "are you done" - Customer says "im done" - Waitress picks up dirty dish from table - Waitress puts down dirty dish on counter - Waitress cleans dirty dish - Delete dirty dish
  - Waitress says "are you done" - Customer says "yes" - Waitress picks up dirty dish from table - Waitress puts down dirty dish on counter - Waitress cleans dirty dish - Delete dirty dish
  - Waitress picks up dirty dish from table - Waitress says "was that okay sir" - Waitress puts down dirty dish on counter - Waitress cleans dirty dish - Delete dirty dish
  - Waitress eats trash - Watiress says "smile"
  - Waitress picks up dirty dish from table - Waitress puts down dirty dish on counter - Waitress cleans dirty dish - Delete dirty dish
  - Waitress says "let me clear that for you" - Waitress picks up dirty dish from table - Waitress puts down dirty dish on counter
  - Customer says "take this for me" - Waitress picks up dirty dish from table
  - ...

- **Players do silly thing**
  - Waitress sits on vase - Customer touches fruit
  - Customer gives menu - Customer gives vase
  - Waitress eats trash - Customer puts down trash
  - Waitress puts down pan - Customer puts down dirty dish
  - Customer eats fruit - Customer picks up bill
  - Customer gets off wine - Waitress cleans dirty dish - Delete dirty dish - Customer looks at menu
  - ...

Appendix B

System Parameters

The following tables list the important parameters of each of the system components, the values used that produced the empirically best[1] results, and a brief description of what the parameter controls. For specifics on their usage check out the source code, the URL is given in Appendix C. Most of these parameters are passed into the components in a configuration file (which is also included in the source code archive), but some are passed as arguments to the program directly or are hard-coded in.

B.1 Word Clustering Component Parameters

[1] In most cases best was qualitatively evaluated based on the output of that stage. This was because of the time requirements of running output from early stages through the entire system and because later stages might have unforeseen effects on the final performance. See chapter 4 for more details

Clustering Parameters		
Parameter	Value	Description
MAX CLUSTERS	3	The number of clusters an element can be in for the purpose of context similarity updates.
UPDATE INTERVAL	20	The number of affinity propagation iterations between updates of the clusters and similarities.
UPDATE DAMPING FACTOR	0.7	The damping factor (0.0–1.0) for updating similarities. Similar to the damping factor below.
DAMPING FACTOR	0.9	The message passing damping factor used by the affinity propagation algorithm [39].
PREFERENCE FACTOR	1.0	Affinity propagation element preferences [39] will be this factor multiplied by the average similarity. This indirectly determines the number of clusters, lower being fewer, higher being more.
AFFINITY ITERATIONS	2000	The number of iterations to run the affinity propagation algorithm for unless the clusters converge in fewer iterations.
CLUSTER ITERATIONS	1	The number of times to run affinity propagation. Each run starts with the hard-clustering output of the previous run.
CONVERGENCE SIMILARITY	0.1	If the net similarity of affinity propagation [39] changes by less than this in CONVERGENCE ITERATIONS we consider the clustering converged.
CONVERGENCE ITERATIONS	40	Used with the above parameter to test for convergence based on net similarity.
CLUSTER CONVERGENCE ITERATIONS	40	The number of iterations the exemplars must stay the same to be considered converged.
FREQUENT THRESHOLD	1	The minimum number of occurrences an element needs to be clustered.
STARTING FREQUENT THRESHOLD	0	If the frequent threshold is 0 we iteratively decrease the frequent threshold starting with this value and rerun clustering each time.
FREQUENT THRESHOLD INCREMENT	0	The amount to decrease the frequency threshold by each iteration in the case described above.
CLUSTER MEMBERSHIP THRESHOLD	0.0	The minimum assignment an element needs to keep track of being in a cluster. For example, a value of 0.1 means that an element must have a soft assignment of at least 10% to be included in a cluster for the purpose of updating context similarity.

Table B.1: Word clustering parameters. These parameters control word clustering, they were determined from related research and empirically from qualitative evaluation of the word clustering results.

Similarity Computation Parameters		
Parameter	Value	Description
ONLY USE FREQUENT CONTEXTS	true	If true, when generating context features only keep those that are only elements that occur at least CONTEXT FREQUENCY THRESHOLD number of times.
USE CLUSTER FREQUENCY	true	If true, when determining if an element meets the requirement below use the size of its primary cluster instead of the element count.
CONTEXT FREQUENCY THRESHOLD	5	The threshold below which elements are considered infrequent and replaced with the infrequent element symbol in contexts.
CONTEXT SIZE	2	The size of context to use, e.g. a value of 3 mean that contexts are three words total, namely two words before a reference word, two words after a reference word or one word before and one word after.
UNWEIGHTED CONTEXT FACTOR	0.1	The factor by which the flat context (number of contexts shared divided by total number either appears in) is included in the score (non weighted) use 0.0 for normal context.
CONTEXT POWER	0.25	The power the context similarity is raised to before being used in the total similarity. This is used to make the context similarity distribution more similar to the surface similarity distribution.
CONTEXT FACTOR	0.9	The weight of context similarity versus surface similarity in the total similarity measure.
SIMILARITY THRESHOLD	0.03	The minimum similarity a pair of elements must have for it to be included in the similarity matrix input to affinity propagation, otherwise they will never be clustered together. This is used to limit the size of the similarity matrix and speed up the clustering process.

Table B.2: Word similarity parameters. These parameters control the way word similarity is measured. They were determined from related research and empirically from qualitative evaluation of the word clustering results.

Unused Parameters		
Parameter	Value	Description
STOP WORD IDF THRESHOLD	-1.0	The minimum IDF / sqrt(sqrt(TF)) needed to be a non stop word. This is used to automatically generate a stop word list based, which are then ignored in contexts and clustering. This is no longer used but might be worthwhile to pursue in the future.
THRESHOLD FACTOR	2.5	If we are only running frequent words through clustering and assigning infrequent words to clusters afterward. This parameter controls how similar an infrequent word must be to some cluster in order to not be put in its own new cluster.
INFREQUENT SIMILARITY THRESHOLD	0.3	The minimum similarity needed when assigning infrequent words to clusters to put that word in the cluster.
FREQUENT WORDS ONLY	true	Whether or not to assign infrequent words to clusters after frequent words have been clustered.

Table B.3: Unused word parameters. These unused parameters were part of earlier experiments, but they were not used in any of the final experiments.

B.2 Dialogue and Action Clustering Component Parameters

Clustering Parameters		
Parameter	Value	Description
MAX CLUSTERS	1	The number of clusters an element can be in for the purpose of context similarity updates.
UPDATE INTERVAL	100	The number of affinity propagation iterations between updates of the clusters and similarities.
UPDATE DAMPING FACTOR	0.7	The damping factor (0.0–1.0) for updating similarities. Similar to the damping factor below.
DAMPING FACTOR	0.9	The message passing damping factor used by the affinity propagation algorithm [39].
PREFERENCE FACTOR	0.7	Affinity propagation element preferences [39] will be this factor multiplied by the average similarity. This indirectly determines the number of clusters, lower being fewer, higher being more.
AFFINITY ITERATIONS	5000	The number of iterations to run the affinity propagation algorithm for unless the clusters converge in fewer iterations.
CLUSTER ITERATIONS	1	The number of times to run affinity propagation. Each run starts with the hard-clustering output of the previous run.
CONVERGENCE SIMILARITY	0.01	If the net similarity of affinity propagation [39] changes by less than this in CONVERGENCE ITERATIONS we consider the clustering converged.
CONVERGENCE ITERATIONS	50	Used with the above parameter to test for convergence based on net similarity.
CLUSTER CONVERGENCE ITERATIONS	50	The number of iterations the exemplars must stay the same to be considered converged.
FREQUENT THRESHOLD	1	The minimum number of occurrences an element needs to be clustered.
STARTING FREQUENT THRESHOLD	0	If the frequent threshold is 0 we iteratively decrease the frequent threshold starting with this value and rerun clustering each time.
FREQUENT THRESHOLD INCREMENT	0	The amount to decrease the frequency threshold by each iteration in the case described above.
CLUSTER MEMBERSHIP THRESHOLD	0.0	The minimum assignment an element needs to keep track of being in a cluster. For example, a value of 0.1 means that an element must have a soft assignment of at least 10% to be included in a cluster for the purpose of updating context similarity.

Table B.4: Dialogue and action clustering parameters. These parameters control dialogue and action clustering. They were determined from related research and empirically from qualitative evaluation of the clustering results.

Similarity Computation Parameters		
Parameter	Value	Description
ONLY USE FREQUENT CONTEXTS	true	If true, when generating context features only keep those that are only elements that occur at least CONTEXT FREQUENCY THRESHOLD number of times.
USE CLUSTER FREQUENCY	true	If true, when determining if an element meets the requirement below use the size of its primary cluster instead of the element count.
CONTEXT FREQUENCY THRESHOLD	10	The threshold below which elements are considered infrequent and replaced with the infrequent element symbol in contexts.
CONTEXT SIZE	2	The size of context to use, e.g. a value of 3 mean that contexts are three words total, namely two words before a reference word, two words after a reference word or one word before and one word after.
UNWEIGHTED CONTEXT FACTOR	0.0	The factor by which the flat context (number of contexts shared divided by total number either appears in) is included in the score (non weighted) use 0.0 for normal context.
CONTEXT POWER	0.5	The power the context similarity is raised to before being used in the total similarity. This is used to make the context similarity distribution more similar to the surface similarity distribution.
CONTEXT FACTOR	0.98	The weight of context similarity versus surface similarity in the total similarity measure.
SIMILARITY THRESHOLD	0.05	The minimum similarity a pair of elements must have for it to be included in the similarity matrix input to affinity propagation, otherwise they will never be clustered together. This is used to limit the size of the similarity matrix and speed up the clustering process.
IDENTICAL THRESHOLD	0.9	The minimum surface similarity needed for two dialogue lines to be considered identical in terms of the clusters their words are in.

Table B.5: Dialogue and action similarity parameters. These parameters control the way dialogue and action similarities are measured. They were determined from related research and empirically from qualitative evaluation of the clustering results.

B.3 Sequence Mining and Filtering Parameters

Parameter	Value	Description
MINIMUM OCCURRENCE COUNT	5	The minimum number of times a sequence must occur.
MAXIMUM LENGTH	7	The maximum number of dialogue lines and actions that the sequence can have.
MINIMUM TRANSITION PROBABILITY	0.01	The minimum transition probability required between adjacent elements of the sequence.
MAXIMUM EXPANSION FACTOR	1.7	The maximum factor by which the average log length of a sequence (the distance between the start and end of a sequence instance in a log) can exceed the number of elements in the sequence.

Table B.6: Sequence mining and filtering parameters. These parameters are used in the mining and filtering components. They determine which sequences are ultimately selected as candidate event-sequences. The parameters were determined through investigation of the output and examination of the golden event labellings.

B.4 Sequence Labeling Parameters

Unused Parameters		
Parameter	Value	Description
MAX SEQUENCE JUMP	5	The maximum distance allowed between consecutive game-actions in a sequence while labeling.
PERCENTAGE COVERED FACTOR	0.95	The percentage of the total labeling quality score that comes from the percentage of the log that is covered by the labeling. The rest of the score comes from the average deviation of the sequence expansion factor (see section 3.8).

Table B.7: Sequence labeling parameters. These parameters control the process of labeling logs with the discovered candidate event-sequences. They were determined from theoretical consideration of what makes for a good labeling and qualitative and quantitative (in terms of the overall system performance) evaluation of the results.

B.5 Sequence Clustering Parameters

Clustering Parameters		
Parameter	Value	Description
MAX CLUSTERS	2	The number of clusters an element can be in for the purpose of context similarity updates.
UPDATE INTERVAL	20	The number of affinity propagation iterations between updates of the clusters and similarities.
UPDATE DAMPING FACTOR	0.8	The damping factor (0.0–1.0) for updating similarities. Similar to the damping factor below.
DAMPING FACTOR	0.9	The message passing damping factor used by the affinity propagation algorithm [39].
PREFERENCE FACTOR	1.0	Affinity propagation element preferences [39] will be this factor multiplied by the average similarity. This indirectly determines the number of clusters, lower being fewer, higher being more.
AFFINITY ITERATIONS	1500	The number of iterations to run the affinity propagation algorithm for unless the clusters converge in fewer iterations.
CLUSTER ITERATIONS	1	The number of times to run affinity propagation. Each run starts with the hard-clustering output of the previous run.
CONVERGENCE SIMILARITY	0.01	If the net similarity of affinity propagation [39] changes by less than this in CONVERGENCE ITERATIONS we consider the clustering converged.
CONVERGENCE ITERATIONS	50	Used with the above parameter to test for convergence based on net similarity.
CLUSTER CONVERGENCE ITERATIONS	50	The number of iterations the exemplars must stay the same to be considered converged.
FREQUENT THRESHOLD	1	The minimum number of occurrences an element needs to be clustered.
STARTING FREQUENT THRESHOLD	0	If the frequent threshold is 0 we iteratively decrease the frequent threshold starting with this value and rerun clustering each time.
FREQUENT THRESHOLD INCREMENT	0	The amount to decrease the frequency threshold by each iteration in the case described above.
CLUSTER MEMBERSHIP THRESHOLD	0.0	The minimum assignment an element needs to keep track of being in a cluster. For example, a value of 0.1 means that an element must have a soft assignment of at least 10% to be included in a cluster for the purpose of updating context similarity.

Table B.8: Sequence clustering parameters. These parameters control sequence clustering. They were determined from related research and empirically from qualitative and quantitative (in terms of the overall system performance) evaluation of the clustering results.

Similarity Computation Parameters		
Parameter	Value	Description
ONLY USE FREQUENT CONTEXTS	true	If true, when generating context features only keep those that are only elements that occur at least CONTEXT FREQUENCY THRESHOLD number of times.
USE CLUSTER FREQUENCY	true	If true, when determining if an element meets the requirement below use the size of its primary cluster instead of the element count.
CONTEXT FREQUENCY THRESHOLD	2	The threshold below which elements are considered infrequent and replaced with the infrequent element symbol in contexts.
UNWEIGHTED CONTEXT FACTOR	0.1	The factor by which the flat context (number of contexts shared divided by total number either appears in) is included in the score (non weighted) use 0.0 for normal context.
CONTEXT POWER	0.25	The power the context similarity is raised to before being used in the total similarity. This is used to make the context similarity distribution more similar to the surface similarity distribution.
NEAR SEQUENCE DISTANCE	3	The maximum number of actions or dialogue lines between the end of one sequence and the beginning of another for them still to be considered near each other. This is used to create the near sequence context, see section 3.9 for more information.
CONTEXT FACTOR	0.85	The weight of context similarity in the total similarity measure.
FEATURE FACTOR	0.1	The weight of the feature similarity (see section 3.9) in the total similarity measure. Surface similarity will have a weight of 1 - CONTEXT FACTOR - FEATURE FACTOR.
SIMILARITY THRESHOLD	0.0	The minimum similarity a pair of elements must have for it to be included in the similarity matrix input to affinity propagation, otherwise they will never be clustered together. This is used to limit the size of the similarity matrix and speed up the clustering process.

Table B.9: Sequence similarity parameters. These parameters control how sequence similarity is measured. They were determined from related research and empirically from qualitative and quantitative (in terms of the overall system performance) evaluation of the clustering results.

Appendix C

Source Code and Other Resources

The source code, project files and external DLLs used have been archived and can be found at `http://web.mit.edu/~tssmith/www/thesis_source.zip`

The official Restaurant Game Project page, with links to the game and research related to the game can be found at `http://web.media.mit.edu/~jorkin/restaurant/`

We are currently gathering data from our newest game, Improviso, which we hope to apply our techniques to in the future. The Improviso home page can be found at `http://gambit.mit.edu/loadgame/improviso.php`

Harley's thesis, which is unrelated to this thesis beyond the fact that this thesis blows Harley's thesis out of the water, can be found with the other CS & EE MEng theses at MIT.

Bibliography

[1] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. LIX, pp. 433–460, 1950.

[2] A. M. Olney, "Dialogue generation for robotic portraits," in *5th Workshop on knowledge and reasoning in practical dialogue systems*, pp. 15–21, IJCAI, January 2007.

[3] "Home page of the Loebner prize in Artificial Intelligence." `http://www.loebner.net/Prizef/loebner-prize.html`, July 2011.

[4] S. Gandhe and D. R. Traum, "First steps towards dialogue modeling from an un-annotated human-human corpus," in *5th Workshop on knowledge and reasoning in practical dialogue systems*, pp. 22–27, IJCAI, January 2007.

[5] M. Mateas and A. Stern, "Procedural authorship: A case-study of the interactive drama Façade," in *Digital Arts and Culture (DAC)*, 2005.

[6] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. London, United Kingdom: Pearson Education, second ed., 2009.

[7] J. Weizenbaum, "ELIZA—a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, pp. 36–45, January 1966.

[8] K. M. Colby, "Modeling a Paranoid Mind," *Behavioural and Brain Sciences*, vol. 4, pp. 515–560, 1981.

[9] R. C. Schank, *Dynamic memory revisited*. New York, NY, USA: Cambridge University Press, 1999.

[10] T. Lane and C. E. Brodley, "Sequence matching and learning in anomaly detection for computer security," in *In Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, pp. 43–49, 1997.

[11] R. Barzilay and M. Lapata, "Modeling local coherence: an entity-based approach," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, (Stroudsburg, PA, USA), pp. 141–148, Association for Computational Linguistics, 2005.

[12] J. Jackson, "Ibm watson vanquishes human jeopardy foes." http://www.pcworld.com/businesscenter/article/219893/ibm_watson_vanquishes_human_jeopardy_foes.html, February 17 2011.

[13] B. Zimmer, "Is it time to welcome our new computer overlords?." http://www.theatlantic.com/technology/archive/2011/02/is-it-time-to-welcome-our-new-computer-overlords/71388/, February 17 2011.

[14] J. Orkin and D. Roy, "Automatic learning and generation of social behavior from collective human gameplay," in *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2009.

[15] S. Ontañón, K. Mishra, N. Sugandh, and A. Ram, "Case-based planning and execution for real-time strategy games," in *Proceedings of the 7th international conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, ICCBR '07, (Berlin, Heidelberg), pp. 164–178, Springer-Verlag, 2007.

[16] J. Orkin and D. Roy, "The restaurant game: Learning social behavior and language from thousands of players online," *Journal of Game Development*, vol. 3, no. 1, pp. 39–60, 2007.

[17] J. Orkin, T. Smith, H. Reckman, and D. Roy, "Semi-automatic task recognition for interactive narratives with EAT & RUN," in *Proceedings of the 3rd Intelligent Narrative Technologies Workshop at the 5th International Conference on Foundations of Digital Games (FDG)*, 2010.

[18] J. McGonigal, "Ted — talks — jane mcgonigal: Gaming can make a better world." http://www.ted.com/talks/jane_mcgonigal_gaming_can_make_a_better_world.html.

[19] J. Orkin, "The restaurant game project." http://www.test.org/doe/, July 2011.

[20] "Improviso." http://gambit.mit.edu/loadgame/improviso.php, 2011.

[21] N. R. Mabroukeh and C. I. Ezeife, "A taxonomy of sequential pattern mining algorithms," *ACM Comput. Surv.*, vol. 43, pp. 3:1–3:41, November 2010.

[22] T. Lane and C. E. Brodley, "Temporal sequence learning and data reduction for anomaly detection," *ACM Transactions on Information and System Security*, vol. 2, pp. 295–331, August 1998.

[23] T. Lau, "A comparison of sequence-learning approaches: Implications for intelligent user interfaces." University of Washington Generals Examination, February 1999.

[24] P. Laird and R. Saul, "Discrete sequence prediction and its applications," in *Machine Learning*, vol. 15, pp. 43–68, 1994.

[25] A. Chotimongkol and A. I. Rudnicky, "Acquiring domain-specific dialog information from task-oriented human-human interaction through an unsupervised learning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, (Stroudsburg, PA, USA), pp. 955–964, Association for Computational Linguistics, 2008.

[26] N. Chambers and D. Jurafsky, "Unsupervised learning of narrative schemas and their participants," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, (Stroudsburg, PA, USA), pp. 602–610, Association for Computational Linguistics, 2009.

[27] J. O. Hilke Reckman and D. Roy, "Learning meanings of words and constructions, grounded in a virtual game," in *Proceedings of the 10th Conference on Natural Language Processing (KONVENS)*, 2010.

[28] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proceedings of the 17th International Conference on Data Engineering*, pp. 215–224, IEEE Computer Society, 2001.

[29] Y. Lu and C. Ezeife, "Position coded pre-order linked WAP-tree for web log sequential pattern mining," in *Proceedings of the 7th Pacific-Asia conference on Advances in knowledge discovery and data mining*, PAKDD'03, (Berlin, Heidelberg), pp. 337–349, Springer-Verlag, 2003.

[30] M. J. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 42, pp. 31–60, 2001.

[31] R. A. Garca-Hernndez, J. F. M. Trinidad, and J. A. Carrasco-Ochoa, "Finding maximal sequential patterns in text document collections and single documents.," *Informatica (Slovenia)*, vol. 34, no. 1, pp. 93–101, 2010.

[32] K. P. Murphy, *Dynamic Bayesian networks: Representation, inference and learning*. PhD thesis, UC Berkeley, 2002.

[33] S. Liang, S. Fuhrman, and R. Somogyi, "Reveal, a general reverse engineering algorithm for inference of genetic network architectures.," *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pp. 18–29, 1998.

[34] K. Murphy and S. Mian, "Modelling gene expression data using dynamic Bayesian networks," 1999.

[35] N. Friedman, K. Murphy, and S. Russell, "Learning the structure of dynamic probabilistic networks," pp. 139–147, Morgan Kaufmann, 1998.

[36] K. P. Murphy, "Bayes Net Toolbox for Matlab." http://people.cs.ubc.ca/~murphyk/Software/BNT/bnt.html, December 2009.

[37] N. Friedman and G. Elidan, "LibB for Windows/Linux programs." http://compbio.cs.huji.ac.il/LibB/programs.html, December 2009.

[38] S. D. LLC., "High performance structure learning: Learning about data relationships." http://www.structureddata01.com/, December 2009.

[39] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, February 2007.

[40] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems* (T. Dietterich, S. Becker, and Z. Ghahramani, eds.), pp. 849–856, MIT Press, 2001.

[41] L. Grönqvist and M. Gunnarsson, "A method for finding word clusters in spoken language," in *Proceedings for Corpus Linguistics*, pp. 265–273, March 2003.

[42] F. Dellaert, "The expectation maximization algorithm," Tech. Rep. GIT-GVU-02-20, Georgia Institute of Technology, February 2002.

[43] J. R. Curran, "From distributional to semantic similarity," tech. rep., The University of Edinburgh, 2003.

[44] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone," in *Proceedings of the 5th annual international conference on Systems documentation*, SIGDOC '86, (New York, NY, USA), pp. 24–26, ACM, 1986.

[45] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, (Stroudsburg, PA, USA), pp. 133–138, Association for Computational Linguistics, 1994.

[46] C. Leacock, G. A. Miller, and M. Chodorow, "Using corpus statistics and wordnet relations for sense identification," *Comput. Linguist.*, vol. 24, pp. 147–165, March 1998.

[47] A. Budanitsky and G. Hirst, "Semantic Distance in WordNet: An Experimental, Application-oriented Evaluation of Five Measures," in *IN WORKSHOP ON WORDNET AND OTHER LEXICAL RESOURCES, SECOND MEETING OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, 2001.

[48] G. Nenadić, I. Spasić, and S. Ananiadou, "Automatic discovery of term similarities using pattern mining," in *COLING-02 on COMPUTERM 2002: second international workshop on computational terminology - Volume 14*, COMPUTERM '02, (Stroudsburg, PA, USA), pp. 1–7, Association for Computational Linguistics, 2002.

[49] G. Nenadić, I. Spasić, and S. Ananiadou, "Term clustering using a corpus-based similarity measure," in *Proceedings of the 5th International Conference on Text, Speech and Dialogue*, TSD '02, (London, UK), pp. 151–154, Springer-Verlag, 2002.

[50] "Mining term similarities from corpora," vol. 10.

[51] D. R. Radev, H. Jing, M. Styś, and D. Tam, "Centroid-based summarization of multiple documents," *Inf. Process. Manage.*, vol. 40, pp. 919–938, November 2004.

[52] D. Wang, T. Li, S. Zhu, and C. Ding, "Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, (New York, NY, USA), pp. 307–314, ACM, 2008.

[53] H. Zha, "Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, (New York, NY, USA), pp. 113–120, ACM, 2002.

[54] J. Yang and W. W. 0010, "CLUSEQ: Efficient and effective sequence clustering," in *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India* (U. Dayal, K. Ramamritham, and T. M. Vijayaraman, eds.), pp. 101–112, IEEE Computer Society, 2003.

[55] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, pp. 832–843, November 1983.

[56] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[57] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *J. ACM*, vol. 21, pp. 168–173, January 1974.

[58] C. I. Ezeife, Y. Lu, and Y. Liu, "PLWAP sequential mining: open source code," in *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, OSDM '05, (New York, NY, USA), pp. 26–35, ACM, 2005.

[59] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 410–420, June 2007.

[60] R. Reichart and A. Rappoport, "The nvi clustering evaluation measure," in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, (Stroudsburg, PA, USA), pp. 165–173, Association for Computational Linguistics, 2009.

[61] A. Bagga and B. Baldwin, "Entity-based cross-document coreferencing using the vector space model," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, (Stroudsburg, PA, USA), pp. 79–85, Association for Computational Linguistics, 1998.

[62] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Inf. Retr.*, vol. 12, pp. 461–486, August 2009.