A Commonsense Approach to Story Understanding

by

Bryan Michael Williams

B.S., Massachusetts Institute of Technology (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

© Massachusetts Institute of Technology 2017. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
May 26, 2017

Certified by .....
Patrick Henry Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor

Certified by .....
Henry Lieberman
Research Scientist
Thesis Supervisor

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

A Commonsense Approach to Story Understanding

by

Bryan Michael Williams

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 2017, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Story understanding is a essential part of human intelligence. Therefore, any complete model of human intelligence must be able to understand stories. Understanding stories requires commonsense knowledge. Many story understanding systems rely on rules to express and manipulate common sense, but the amount of common sense is vast and expressing significant amounts of common sense this way is tedious. Commonsense knowledge bases can help address this problem. My work demonstrates the power of combining a story-understanding system and a commonsense knowledge base.

Genesis is a story-understanding system that models aspects of story understanding by reading and analyzing stories written in simple English. I have extended Genesis by incorporating ConceptNet, a large knowledge base of common sense, into its processing. Now, Genesis not only has access to the dozens of inference rules provided by Genesis users, but also tens of thousands of goal-related assertions provided by contributors to ConceptNet. Genesis uses ConceptNet's knowledge to infer characters' goals, in some cases allowing Genesis to understand stories with few or even zero user-defined rules. Genesis can now infer relevant events and connections never explicitly stated in the story in a manner unprecedented within the space of story understanding and machine reading. Genesis also uses ConceptNet to apply user-defined rules in a commonsense, more flexible fashion, extending the reach of a rule by factors ranging from ten to one hundred. These additions to the Genesis story-understanding system takes steps toward practical AI-enabled systems that provide advice in areas that heavily depend on precedent, including law, medicine, and business.

Thesis Supervisor: Patrick Henry Winston
Title: Ford Professor of Artificial Intelligence and Computer Science

Thesis Supervisor: Henry Lieberman
Title: Research Scientist

Acknowledgments

Thank you to the Genesis group, which has been a regular source of support and inspiration for two years.

Thank you to my family for all of their support and encouraging me to do the M.Eng program.

Thank you to my lovely girlfriend Sidni for everything she does.

Henry Lieberman was an essential part of this project, providing the seeds for much of the work that follows. Thank you for making me a more ambitious, directed, and confident researcher.

Patrick Winston has taught me how to write, speak, and think. He has provided both great freedom yet also necessary guidance. I am very grateful to have worked so closely with him.

This research was supported, in part, by the Air Force Office of Scientific Research, Award Number FA9550-17-1-0081.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "Only a small community has concentrated on general intelligence. No one has tried to make a thinking machine and then teach it chess - or the very sophisticated oriental board game Go...The bottom line is that we really haven't progressed too far toward a truly intelligent machine. We have collections of dumb specialists in small domains; the true majesty of general intelligence still awaits our attack."
>
> —Marvin Minsky (Stork, 1997)

1.1 Overview

Story understanding is essential to artificial general intelligence. Here, I use the term "story" to refer to any related sequence of events; a traditional narrative structure is not necessary. Humans communicate in stories, and the ability to fully understand human communication would enhance all aspects of a computer's interaction with human text, speech, or even thoughts. Better machine readers, question answers, and intelligent assistants are only the beginning. Although today's popular approaches to natural language tend to ignore commonsense knowledge, it is a vital component of story understanding. "Commonsense knowledge" here broadly refers to all of the everyday, obvious knowledge humans hold and often keep implicit when

communicating—ordinary facts like wings are used for flight and students typically desire good grades. Many story understanding systems rely on rules to express and manipulate common sense, but the amount of common sense is vast and expressing significant amounts of common sense this way is tedious. Commonsense knowledge bases can help address this problem.

Genesis is a story-understanding system whose rule-based approach to story understanding provides humanlike flexibility. Genesis is able to analyze a story and demonstrate its understanding in numerous ways, including question answering, summarization, hypothetical reasoning, and story alignment (Holmes & Winston, 2016; Winston, 2014). Its rule-based core and general knowledge representation are used to complete all these tasks and many others. However, its reliance on rules sacrifices breadth. Prior to my work, the user had to explicitly give Genesis commonsense rules that codify all the knowledge necessary for identifying connections between story events.

I have tackled the problem of common sense enabled story understanding by connecting Genesis to ConceptNet, a large commonsense knowledge base. Equipped with access to hundreds of thousands of commonsense assertions, Genesis can now analyze text in novel ways, such as using its background knowledge to infer implicit explanations for story events. These explanations can reference events never explicitly stated in the story. Genesis detects that these inferred explanations are relevant using the context its common sense provides. Moreover, Genesis provides a human-readable justification for every common-sense-assisted inference it makes, showing the user exactly what pieces of commonsense knowledge it believes are relevant to the story situation. ConceptNet's substantial size allows these features to function at a large scale, and its knowledge is general enough to be widely relevant, yet descriptive enough to allow for nontrivial inferences. Inferring nontrivial implicit events at a large scale is a new capability within the space of machine reading and story understanding in both rule-based and machine learning approaches.

Connecting Genesis and ConceptNet demonstrates the power of combining a story understanding system with a commonsense knowledge base. The user or programmer

is no longer required to specify all the commonsense knowledge Genesis should employ because Genesis contacts ConceptNet to retrieve commonsense knowledge relevant to the current story analysis. Genesis now uses both user-defined rules and ConceptNet knowledge together to analyze a story. While both representations express commonsense knowledge, ConceptNet's large size makes it a valuable repository of common sense that gives Genesis a great breadth of knowledge and reduces the human burden of supplying this knowledge. My work enables Genesis to directly apply ConceptNet knowledge that forms the equivalent of 30,000 rules. I also developed a way that Genesis can apply user-defined rules in a commonsense, more flexible fashion with the help of ConceptNet, making them approximately ten to one hundred times more powerful.

ConceptNet contains approximately 225,000 commonsense assertions. Although my work does not enable Genesis to explicitly use every type of ConceptNet's knowledge in all aspects of story processing, the connection I have established between the systems is highly general and extensible. Therefore, the architecture supports incorporating new kinds of ConceptNet assertions, enabling Genesis programmers to add to the thousands of assertions currently accessed.

To show the value of leveraging a commonsense knowledge base in a story-understanding system, consider the following example story describing the Mexican-American war, a conflict which occurred in the 1840s:

> The United States is a country. Mexico is a country. The year is 1846. Manifest Destiny is popular in the United States. The United States has ambition, has greed, and wants to gain land. Mexico and the United States disagree over borders. The United States move into the disputed territory. Mexican forces attack the United States. The United States declares war on Mexico. Winfield Scott leads the United States army. The United States battles Mexico. The United States triumphs by capturing Mexico City. The United States defeats Mexico and wins the war. The Treaty of Guadalupe Hidalgo officially ends the war.

This summary of the war is plain, brief, and high-level, but resembles an elementary school textbook chapter summary. When Genesis analyzes this story with the help of ConceptNet, it infers the United States' goals from the explicit story events that cause and contribute to these goals. Genesis performed this analysis without any user-defined rules, relying only on ConceptNet knowledge. ConceptNet helps Genesis connect the United States' ambition to its goal of conquering its opponent, which it completes when it triumphs by capturing Mexico City. Similarly, Genesis infers that the United States conquers a nation by winning the war, and its goal of conquering a nation stemmed from its greed. Genesis concludes that the United States gains land by conquering. Note that the story does not explicitly state any causal connections, so all causal connections Genesis forms are due to the application of commonsense knowledge.

We can explore some of these inferences by asking Genesis questions, as shown in Figure 1.1. When asked "How did the United States gain land?", Genesis responds with two story elements which were never explicitly stated in the story, but which it inferred using its common sense. When questioned about some of these inferred elements, it responds with further explanation, citing explicit story events to back up its inferences. The language "conquer opponent" and "conquer nation" does not appear in the story text, but Genesis has inferred that these terms are relevant and has connected these inferences to explicit story events. The grammar is not perfect, and Genesis's conclusions are just one interpretation of the story's events and depend on its commonsense knowledge. However, Genesis's effective application of relevant knowledge, not the knowledge itself or the English generation, is the focus of my work and the example.

Supplying a story-understanding system with large amounts of commonsense knowledge allows it to comprehend its input more fully, approaching the understanding that humans have when reading the same input. Commonsense knowledge enables Genesis to make inferences that it would otherwise miss, including inferences that contain events and connections never explicitly stated in the story. With the additional commonsense knowledge ConceptNet provides, the ease of applying Genesis's analysis

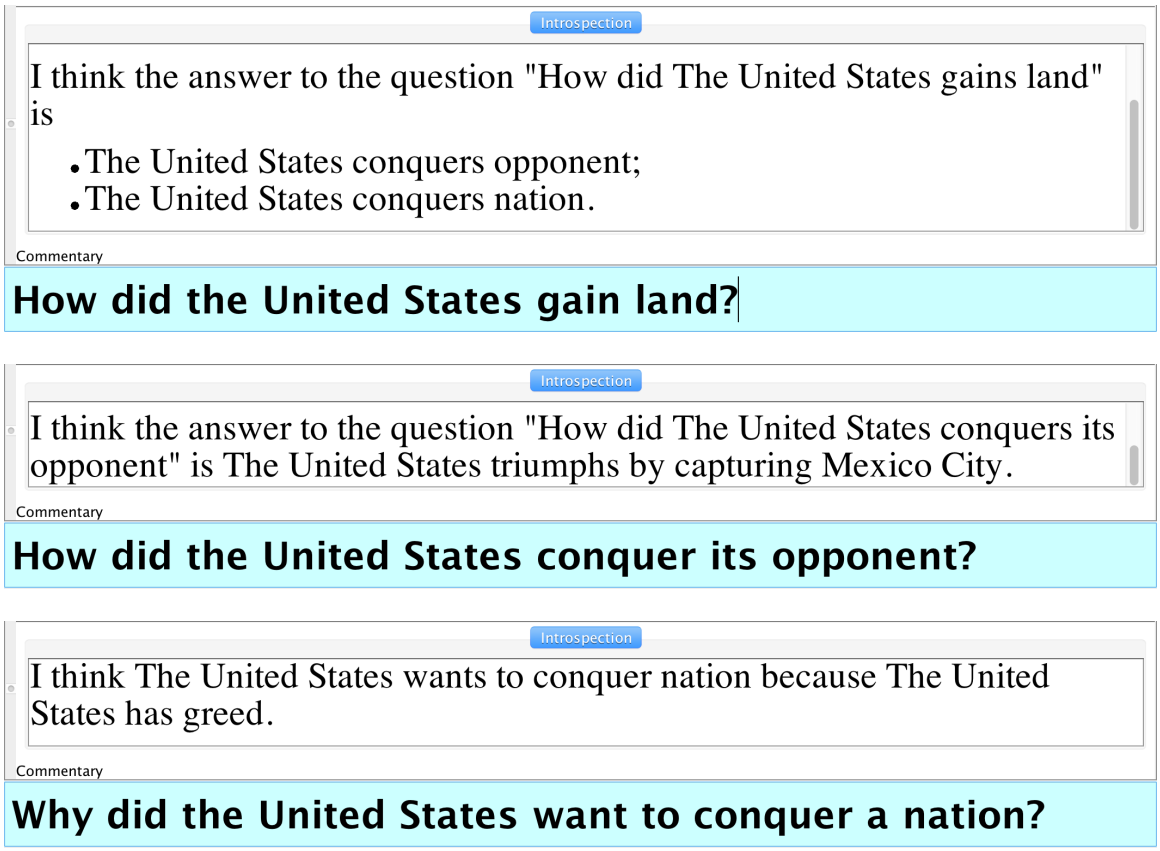

Figure 1.1: A series of questions and Genesis answers regarding the Mexican war story. Genesis answers the first question with two story elements which were never explicitly stated, but which it inferred using its common sense. The second and third questions ask Genesis about these inferences, and it backs them up with explicit story elements. Genesis's grammar in its English generation is not perfect, but the content is salient and the inferences are interesting.

to a large corpus of naturally occurring text increases significantly. In the remainder of this introductory chapter, I examine the larger vision, my exact approach, and my specific contributions.

1.2 Vision: Artificial General Intelligence

The development of artificial general intelligence, also known as strong artificial intelligence, aspires to produce a computational system that can meet or surpass the capabilities of humans in all or nearly all aspects of reasoning and intelligent behavior. This is in contrast with weak artificial intelligence, which focuses on designing

systems to solve problems restricted to specific domains. Many of the recent, popular advances in artificial intelligence (AI) fall under weak AI, including Siri, AlphaGo, and IBM Watson. These systems demonstrate extremely impressive abilities in their respective domains, and Watson and AlphaGo perform better than even the most expert humans. However, it is not clear how much these systems contribute to the goal of strong artificial intelligence because of their narrow domain and limited knowledge representation (Ferrucci et al., 2010; Levy, 2017; Silver et al., 2016). The knowledge representation, a way of representing information about the world, is a crucial part of an AI's generality.

These systems have been built to tackle a very narrow range of problems. While these problems are certainly difficult, the limited scope results in a high degree of engineering specialization. For instance, Siri can easily complete the task of showing me my friend Chase's contact information if I request it. However, if I ask Siri when I last talked to Chase, the assistant is unable to respond with a useful answer even though my phone and computer keep track of this information. Similarly, Siri on my Mac laptop can open my thesis document upon request without issue, but cannot answer the question of when I last changed my thesis. An example of an unsatisfactory interaction is shown in Figure 1.2. It's difficult to get the assistant to complete related tasks even when it has the necessary information because its knowledge representation is extremely shallow. Apple is secretive regarding the algorithms behind the system, but it seems as though Siri is completing rudimentary pattern matching rather than performing a deeper analysis of requests, their logical components, and how they relate to device data.

These limitations are common in large machine learning and natural language processing systems. Machine learning is generally focused on specific lower-level classification and regression tasks rather than higher-level reasoning and problem solving (Langley, 2011). Because these systems are usually designed to complete one sort of task, the knowledge representation is lightweight and lacks common sense. Moreover, knowledge used to solve one problem often cannot be used to solve a related but slightly different problem. Humans, by comparison, are extremely flexible and

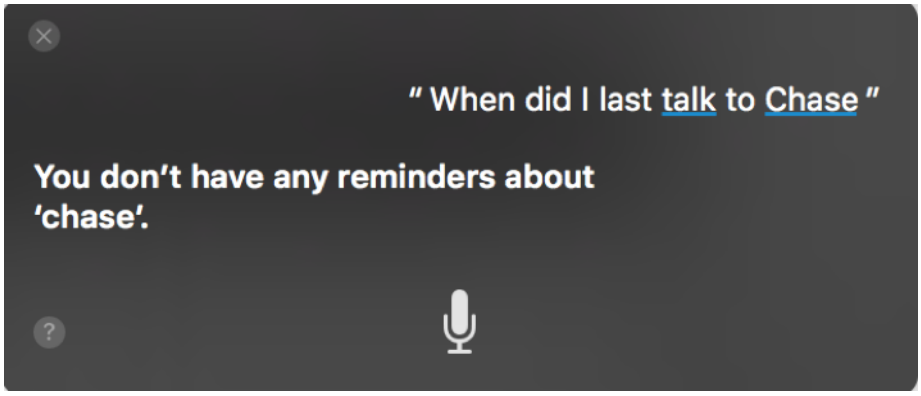

Figure 1.2: An example of an unsatisfying interaction with Siri. Siri is able to complete many specific tasks, but lacks the flexibility and common sense to complete even small variations of these tasks.

quick learners. We excel at generalization, abstraction, and analogy, allowing for nearly effortless knowledge transfer (Gentner & Markman, 1997). While there has been some recent research on applying a neural network trained for one problem to a related problem, the problem domain is limited to perception. It's not clear that such an approach could be used on higher-level problem-solving tasks (Oquab, Bottou, Laptev, & Sivic, 2014). Humans rely on a flexible knowledge representation to identify causal connections, learn from a small number of examples, and transfer knowledge between domains, all vital for generally intelligent behavior. Therefore, the knowledge representation is clearly of crucial importance in building a strong AI system.

1.3 The Importance of Common Sense

> "To solve the hard problems in AI — natural language understanding, general vision, completely trustworthy speech and handwriting recognition — we need systems with common sense knowledge and flexible ways to use it. The trouble is that building such systems amounts to 'solving AI.' This notion is difficult to accept, but it seems that we have no choice but to face it head on."
>
> —Push Singh, co-founder of the Open Mind Common Sense project

(Singh, 1996)

However, the knowledge representation is not the whole story. Consider designing a language understanding system and feeding it the following example story from a dataset on teen bullying (*MTV Over the Line*, n.d.):

> "My bf keeps asking where i am, what i am doing and who i am with!! He wont let me hang out with any guys even if it is for a school project and when i dont answer he tries to track me by facebook and and [sic] by my friends. Is that obsesive [sic]?????!!!!!!"

Ideally, the system would be able to understand the user's input and respond with helpful advice. An adult human reader is easily able to comprehend the story and identify how the events are connected. However, slang and grammatical errors aside, reaching this understanding requires a stunning amount of commonsense reasoning so automatic in human cognition that it's rarely apparent. The reader must realize that

- it's common for a person's friend to know that person's whereabouts to understand why the subject's boyfriend was interacting with the subject's friends,
- completing a project oftentimes requires spending time together to understand why the subject included the detail about the school project, and
- behavior such as tracking a partner's location and controlling his or her social life are signs of an abusive relationship.

Other more mundane pieces of common sense, such as the knowledge that "hanging out" is a form of social interaction and Facebook is a social network containing geolocation data, are also required, but the three points listed directly above are more essential though perhaps less visible to humans. All of this commonsense knowledge is "fuzzy"—it's not always true that an abusive boyfriend tracks his partner's location, or that a boyfriend tracking his partner's location is abusive. Such loose restrictions

on this knowledge make it difficult to computationally pin down. However, humans employ this sort of commonsense knowledge routinely when communicating and processing information. Therefore, any strong artificial intelligence designed to interact with human speech, human writing, or even human minds must have some mechanism for representing and operating upon commonsense knowledge. I will refer to such an AI as common-sense-friendly.

Machine learning approaches are certainly useful and incredibly strong at pattern recognition and similar tasks. However, these systems are often too narrow to contribute to the goals of strong artificial intelligence on their own. Systems with a general knowledge representation that incorporates common sense are necessary for higher-level reasoning and more directly align with the goals of common-sense-friendly strong AI. My work takes steps towards the realization of such a system.

1.4 Approach

I tackled the problem of developing a common-sense-friendly artificial general intelligence by extending a story-understanding system with commonsense reasoning capabilities.

Genesis is a rule-based story-understanding platform which can read in any arbitrary sequence of events in simple English (a story) and analyze it in intelligent ways. Some of its features include question answering, hypothetical reasoning, analyzing a story from the perspective of different characters, and detecting analogous events in two different stories (Holmes & Winston, 2016; Winston, 2014). Genesis has a flexible knowledge representation which allows it to store knowledge about arbitrary story events, and its reasoning and problem-solving models are domain-agnostic as well. These appealing qualities are present because Genesis was designed to model general story understanding and human problem solving rather than complete any one specific task.

Detecting causal relationships is central to many of Genesis's features. Causal relationships describe how different events in the story are connected, and these con-

nections allow Genesis to understand why the events proceeded in the way they did. Prior to my work, Genesis acquired all of its knowledge about causal relationships from rules users constructed by hand. A user would have to manually tell Genesis that, for instance, if person A harms person B, person B gets upset. Genesis applies these rules while reading the story in order to make inferences and increase understanding. Genesis's dependence on these user-defined rules is one of its most prominent limitations. In order for Genesis to understand a story about a war, for example, a user would have to construct rules that describe the sort of causal relationships present in war-related events. Many of these rules would describe common sense, such as the fact that bigger armies help win wars.

When applying Genesis to a large corpus of "real-world," naturally occurring text rather than carefully constructed example stories, requiring users to anticipate all the necessary commonsense knowledge needed to understand the stories becomes unfeasible. Instead, Genesis should be able to make the commonsense connections on its own, and the user can define rules that discuss the specifics of a particular domain or story. In this way, common sense and domain-specific knowledge are separated, and Genesis can be much more easily applied to large stories about arbitrary topics. Equipped with large amounts of commonsense knowledge, Genesis becomes much more "intelligent" on its own without the user's explicit contributions, and can even make valuable connections within the story that the user did not expect.

ConceptNet is a large knowledge base of commonsense assertions. ConceptNet is organized into concepts, which are simple nouns, verbs, or noun/verb phrases, and relations between these concepts. ConceptNet has many different versions, but the one I made primary use of, ConceptNet 4, features approximately 225,000 assertions describing commonsense knowledge, all submitted directly by humans. I connected Genesis's story processing to ConceptNet so Genesis can use ConceptNet's knowledge while reading a story to make connections it would otherwise miss. ConceptNet is an ideal source of commonsense knowledge because it captures human commonsense knowledge instead of encyclopedic knowledge.

I applied ConceptNet's knowledge within Genesis in two specific ways: common

sense enabled rule matching (CSERM), a method of more flexibly applying user-defined rules to story events using common sense, and ASPIRE, a system for inferring characters' goals from the actions they taken within the story. These two applications are completely separate, illustrating different ways Genesis can take advantage of ConceptNet's knowledge. Both have a big impact. CSERM amplifies the applicability of each rule by a factor of approximately ten to one hundred, making each user-defined rule much more powerful. The ConceptNet knowledge ASPIRE incorporates into its processing is equivalent to approximately 30,000 equivalent Genesis rules. In both applications, Genesis provides justifications to the user explaining the inferences it made with the help of ConceptNet by citing the specific pieces of relevant knowledge.

While CSERM does use ConceptNet's collective knowledge, CSERM only expands existing user-defined rules and cannot stand on its own to aid story processing. Conversely, ASPIRE allows Genesis to leverage thousands of ConceptNet goal-related assertions to make inferences about character goals with no dependence on user-defined rules. My work does not currently incorporate the entirety of ConceptNet's knowledge into all aspects of Genesis processing. Applying ConceptNet knowledge in the standalone manner that ASPIRE does requires programmer diligence and is therefore currently limited to goal-related knowledge. However, the extensible connection I have established between the two systems supports querying for any assertion type of interest, paving the way for additional ConceptNet knowledge to be applied with no dependence on the user.

Much of my work focuses on significantly reducing the number of necessary user-defined commonsense rules. I am not advocating for abandoning Genesis rules, as they are still quite useful and express ideas ConceptNet cannot. Rather, I am trying to show the power of ConceptNet's commonsense knowledge and how much it offers Genesis. Both user-defined rules and ConceptNet assertions express commonsense knowledge, but ConceptNet contains several orders of magnitude more assertions than the dozens of currently constructed Genesis rules. Now, Genesis uses both forms of knowledge together to analyze a story.

With the commonsense reasoning that CSERM and ASPIRE provide, applying

Genesis's analysis to a corpus of naturally occurring text becomes significantly more feasible. Genesis has a rich set of features and analytical tools, and its ConceptNet-assisted commonsense reasoning makes casual connections which allow these capabilities to perform better with minimal user effort. This technology would be very useful in any domain that heavily depends on precedent, including law, medicine, and business. Attractive Genesis features including detecting the similarity between documents, finding higher-level user-defined story patterns such as revenge and Pyrrhic victory, and performing hypothetical reasoning could all help Genesis analyze a corpus to pick out documents or passages of interest (Holmes & Winston, 2016; Winston, 2014). Genesis's analysis capabilities are considerably more nuanced and explainable than comparable machine-learning approaches, and ConceptNet knowledge bolsters their performance and ease of use. Giving Genesis its own common sense means users don't have to spell out as much to Genesis and can instead spend their time describing domain-specific knowledge. My work is a step towards achieving this vision of a widely applicable system for humanlike analysis.

1.5 Contributions

My work in extending Genesis with commonsense knowledge has five primary contributions:

- Established a general, extensible connection between Genesis and ConceptNet which can easily generate user-friendly justifications for inferences
- Implemented common sense enabled rule matching in Genesis, allowing Genesis to amplify its rule set by using ConceptNet to better detect when a rule applies to a story event
- Articulated a framework for inferring the goals of characters from actions taken within the story
- Implemented ASPIRE and incorporated ConceptNet knowledge for detecting

both goal causation and goal completion, enabling Genesis to understand goal-oriented stories with a significantly reduced dependence on user-defined rules

- Demonstrated the feasibility of combining a story-understanding system and a commonsense knowledge base to infer nontrivial relevant events and connections never explicitly stated in the input text at a large scale

1.6 Chapter Descriptions

Chapter 2 gives general background about Genesis, covering its features and limitations at the inception of this work, and Chapter 3 gives background about ConceptNet, discussing the structure of its information and its different versions. Chapter 4 examines the connection between Genesis and ConceptNet I designed and implemented, focusing on its ease of use and extensibility. Chapter 5 explores CSERM, an initial application of ConceptNet's commonsense knowledge in Genesis which makes rules more flexible, and Chapter 6 introduces ASPIRE, a more ambitious application of ConceptNet knowledge focused on character goals. Chapter 7 shows the results of using CSERM and ASPIRE on three example stories and quantifies the amount of common sense Genesis has gained. Chapter 8 discusses some limitations of Genesis's newly accessible common sense and opportunities for future work. Chapter 9 discusses related work, including machine learning approaches to story understanding, prior attempts at giving story-understanding systems common sense, and additional commonsense knowledge bases. Finally, Chapter 10 concludes the thesis and articulates the main contributions of my work.

Chapter 2

Genesis Background

2.1 Overview

Genesis is a story-understanding system started by Patrick Winston in 2008 and developed with the help of many students. Genesis parses natural language input and forms its own internal representation of the story. "Story" here is used to refer to any sequence of events; it is not required to follow a traditional narrative format. At the inception of this work, Genesis obtained common sense solely from user-defined rules. Applying these rules allows Genesis to form causal connections between the story's events. Once Genesis has read the story and formed its internal representation, it can demonstrate its understanding in many ways and perform additional sorts of analysis at the user's request.

Genesis is implemented in Java and uses the Katz's (1997) START parser to assist with reading in the input text. Genesis maps the results of START into its own inner language. This inner language expresses the essential contents of every sentence while omitting unnecessary details. The inner language uses four primary types of representation substrate to capture information within a sentence or phrase:

- Entities, which can be primitive nouns such as "baseball" or "love"
- Relations, which describe connections between entities. An example relation is "Matt harms Josh"

- Functions, which modify an entity and represent things such as prepositional phrases

- Sequences, which allow entities to be grouped together

Importantly, entities can be used to represent specific nouns, but functions, relations, and sequences are all entities as well. Therefore, entity is a general term used to describe Genesis's inner language representation of any sentence or sentence fragment. All story events are entities, but not all entities are story events because entities can represent the smaller components of a story event as well.

Genesis supports adding properties to an entity which describe its attributes. Properties are implemented as a hashmap—every property has a String name, its key, and an Object value. These are useful for keeping track of information about an entity over the course of Genesis's story processing, and I used properties to hold information about why and how an inferred entity was inferred in my common sense enabled rule matching and ASPIRE implementations.

Genesis uses WordNet to form a list of threads which describe the different senses, or glosses, of a word. Each thread contains a full hierarchy describing a chain of categories which the word fits into. For instance, WordNet provides "thing entity physical-entity object whole living-thing organism animal chordate vertebrate mammal placental carnivore canine dog" as a thread for dog. Genesis uses threads when matching story events to rules; if a story has a rule about animals and a story event contains a dog, the dog in the event would match the animal in the rule because the thread for dog tells Genesis that a dog is an animal. However, WordNet's hierarchies are relatively rigid, often not capturing commonsense usage of these words. Note that the hierarchy in the given dog thread does not contain "pet" even though dogs are often pets. ConceptNet helps address some of these limitations, better capturing people's commonsense ideas and associations with these words. WordNet is particularly limited with verb hierarchies, and ConceptNet does well with verbs (Miller, 1995).

Genesis relies heavily on making causal connections between the story's events to perform intelligent analyses on the story. Causal connections enable Genesis to view

the story as more than just a sequence of unrelated events; with these connections, Genesis reaches a much more complete and explicit understanding of the story. The elaboration graph is a central display in Genesis because it prominently features these causal connections. The graph separates story events into a set of boxes and illustrates causal relationships between two events by drawing a line between the corresponding boxes. There is an extensive color scheme for the boxes and lines which describes the types of entity represented by the boxes and the sorts of causal connection represented by the lines, but I omit that here because it's not directly relevant to my work.

2.2 Capabilities

Much effort has gone into giving Genesis a rich set of analytical features. Genesis can summarize stories, answer questions about them, analyze its own problem solving process in the form of a story about itself, perform hypothetical reasoning, identify story patterns in a story from user-defined templates, align two stories to identify analogous events, apply a cultural bias in its interpretation, apply rules specific to character personalities, and analyze a story from the perspective of one of the characters, among other capabilities (Holmes & Winston, 2016; Noss, 2017; Winston, 2014). Many of these features depend on the causal connections which Genesis has made, so if those connections are incomplete or of poor quality, the analysis suffers.

Figure 2.1 shows Genesis's interpretation of a simple English version of Macbeth given user-defined commonsense rules and story pattern templates. The lower pane shows the elaboration graph for the entire story, while the upper pane shows the elaboration graph specific to the story pattern of "Pyrrhic victory" which Genesis has identified in the story.

2.3 Limitations

Genesis is quite powerful, but has two major drawbacks. The first is that its parsing capabilities are somewhat limited. The START parser is used to turn English into

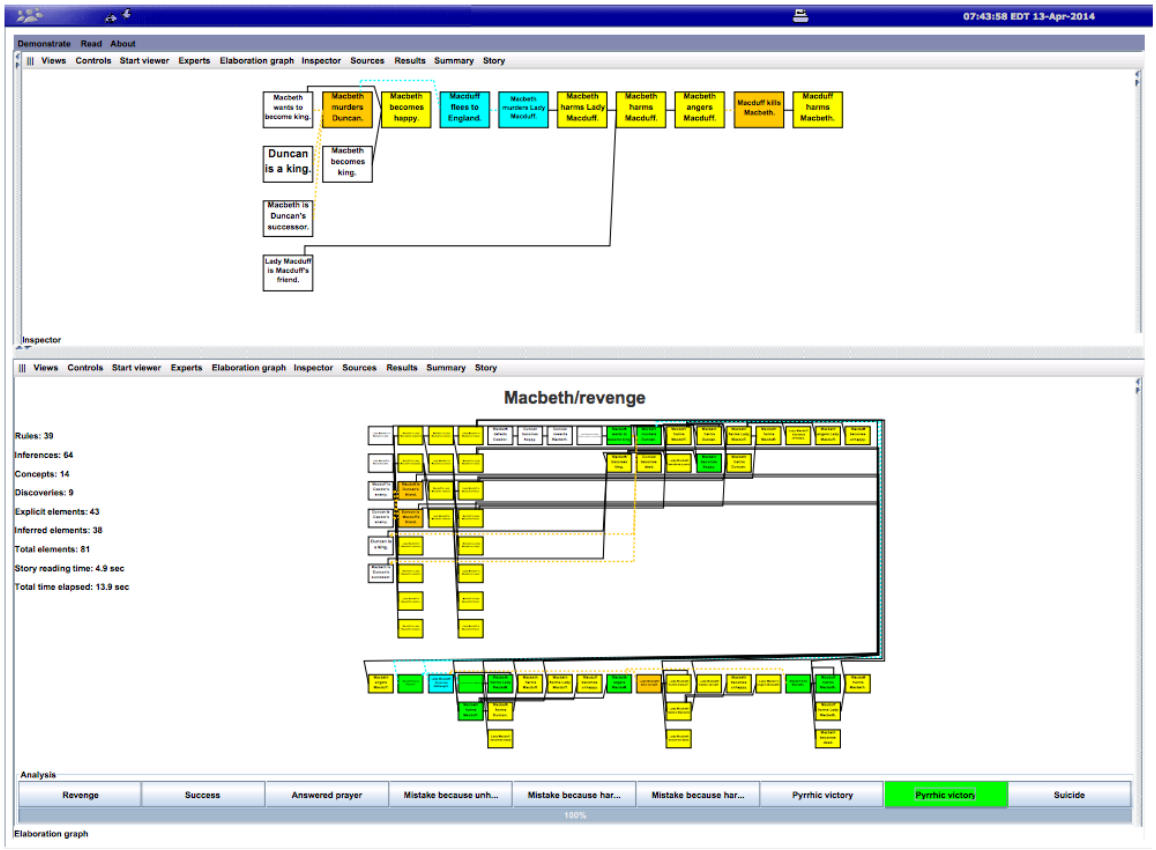

Figure 2.1: A display of two elaboration graphs from Genesis's reading of a Macbeth summary in simple English. The bottom is the full elaboration graph, showing all explicit and inferred story entities, and the top displays just those events relating to the user-defined story pattern of "Pyrrhic Victory." From "The Genesis Story Understanding and Story Telling System: a 21st Century Step Toward Artificial Intelligence," by P. Winston 2014. Reprinted with permission.

Genesis's inner language. START can parse simple sentences without issue, but struggles as the complexity of the sentence rises. Therefore, Genesis can currently only be fed stories written in relatively simple English and cannot be readily applied to a large corpus of naturally occurring text. While this is an important problem, this is not the focus of my work.

Instead, my work targets Genesis's need to be given user-defined rules that describe the necessary commonsense knowledge. This places a large burden on the user of Genesis because the user must tell Genesis banal pieces of common sense. For example, a user could express the fact that people get angry when they're harmed by using Genesis's rule notation to create the rule "If XX harms YY, YY becomes

angry," where XX and YY can match specific people in the input story. If Genesis is to be applied to longer stories or a large corpus of stories, the user would need to express all of the necessary knowledge required to produce useful understanding, which makes using Genesis much less attractive. Genesis is a great tool for automating story analysis, but if the user would need to first read all the stories herself to identify the necessary common sense, the analysis is no longer automated. These rules are of great importance to Genesis because, at the inception of this work, they were the only way that Genesis could make causal connections, and causal connections are vitally important for intelligent story analysis.

The same ruleset is likely useful for many stories, especially within the same genre, so the user may be able to reuse or modify previously constructed rulesets instead of constructing a full ruleset for every story of interest. Still, ConceptNet improves Genesis's usability because it provides access to hundreds of thousands of commonsense assertions, removing the need for many user-defined rules in the first place. ConceptNet provides a common store for commonsense knowledge, and it is larger in size than any previously constructed Genesis ruleset. My work strives to give Genesis part of the common sense necessary for identifying the obvious, standard causal connections that a human reader would make if given the same story. This helps the user focus on giving Genesis domain-specific rules used to express knowledge that isn't common sense. Furthermore, all of the existing story analysis capabilities are bolstered by the commonsense connections that Genesis makes using ConceptNet even in the absence of user-defined rules! Because commonsense knowledge is of great value to Genesis, my work focuses on giving Genesis access to large amounts of this knowledge and facilitating its use in story processing.

Chapter 3

ConceptNet Background

3.1 Overview

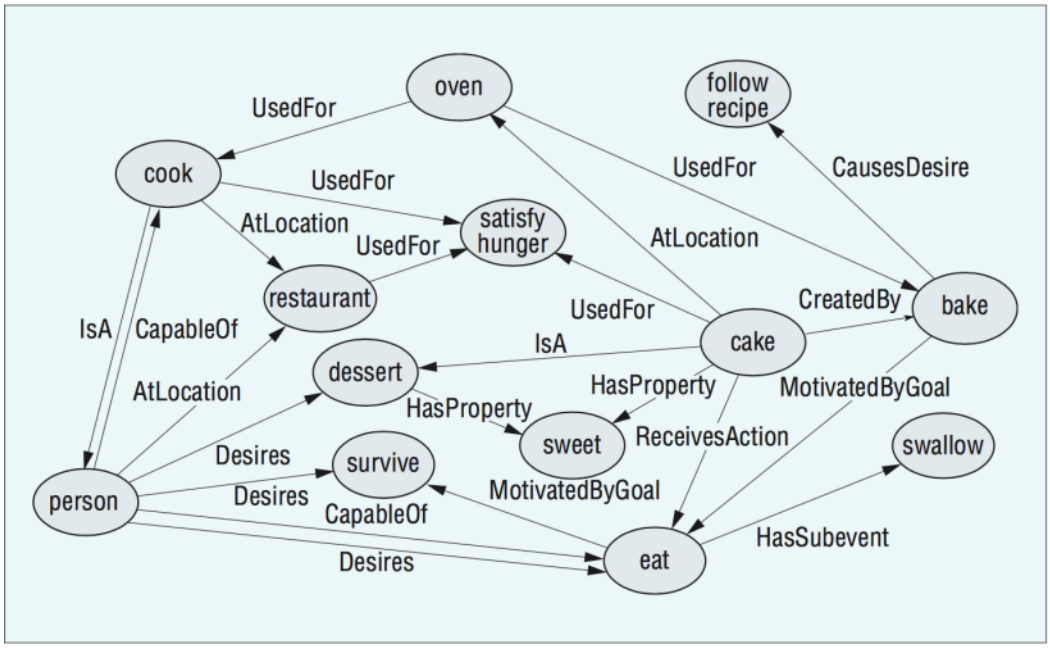

Figure 3.1: A visualization of a very small part of ConceptNet's knowledge. Concepts are shown in circles and relations are shown as directed arrows between concepts. From "Digital intuition: Applying common sense using dimensionality reduction," by C. Havasi, R. Speer, J. Pustejovsky, & H. Lieberman, 2009, IEEE Intelligent Systems, 24, p. 4. Copyright 2009 by IEEE. Reprinted with permission.

ConceptNet is a large semantic network of common sense developed by the Open Mind Common Sense (OMCS) project at MIT. The OMCS project was founded in

1999 with the mission of capturing the commonsense knowledge necessary to enable computers to reason in more humanlike ways. Marvin Minsky, Push Singh, Catherine Havasi, Rob Speer, and Henry Lieberman all were or are notable involved researches. OMCS started with a crowdsourcing approach to data collection, as acquiring commonsense knowledge by other means is not straightforward. OMCS has explored several representations of commonsense knowledge, with ConceptNet being the most influential (Havasi, Speer, Pustejovsky, & Lieberman, 2009; Liu & Singh, 2004; Singh, 2002; Singh, Barry, & Liu, 2004).

There are several versions of ConceptNet, but the representation for commonsense knowledge is shared. ConceptNet represents its knowledge using concepts and relations between them (note that, while user-defined story patterns such as those describing "Pyrrhic victory" are commonly referred to as "concept"s in several other Genesis texts, throughout this thesis I am using "concept" only in the ConceptNet sense of the word). Concepts can be nouns such as "computer", verbs such as "exercise," or short phrases such as "breathe fresh air" or "beanbag chair." Relations are used to describe a connection between two concepts. Some common relations are "Is A," "Used For," "Causes," "Desires," "At Location," and "Has Property." ConceptNet represents its commonsense knowledge using assertions, each of which consists of a left concept, a relation, and a right concept. For instance, the ConceptNet assertion "computer At Location office" represents the fact that computers are commonly found in offices. Every assertion is also associated with a score which represents the confidence in the truth of that assertion. Negative assertions are represented with a negative score (e.g. the assertion "person Desires broken leg" has a score of -1.85 in one version of ConceptNet). A feature is a ConceptNet assertion with one relation missing such as "person Desires ____ " or "____ Used For build boat." Features are useful for asking ConceptNet open-ended questions. For example, "person Desires ____" could be used to ask ConceptNet for all the things it knows that people desire (or don't desire, if the score is negative) (Havasi et al., 2009; Liu & Singh, 2004).

OMCS has introduced many different versions of ConceptNet over the years, with development stretching back to 2000 (Liu & Singh, 2004; Speer & Havasi, 2012).

ConceptNet 4 was introduced in 2008, features multiple languages, and supports a web API which users can query to make use of commonsense knowledge. ConceptNet 4 contains about 22,000 English concepts connected in approximately 225,000 assertions and uses 27 relations. The knowledge in ConceptNet 4 and previous versions was all directly collected from humans. OMCS created a website where users could submit commonsense knowledge, either in free text or by using templates which constrain the format of the submission. The free text was parsed using a simple parser and converted into an assertion. If multiple users submitted the same assertion, the score for that assertion increased (or decreased, if the assertion is negative) to represent the increased confidence. Users could also rate previously submitted assertions, and these ratings influenced the scores of the assertions (Havasi et al., 2009).

ConceptNet 5 was introduced in 2012 and contains much more knowledge than previous versions of ConceptNet in both English and other languages. ConceptNet 5 has around 3.9 million concepts connected in 8.7 million assertions. This version of ConceptNet still contains knowledge collected from humans, but complements that knowledge with information from WordNet, Wikipedia, and several other sources (Speer & Havasi, 2012). It also supports queries through its web API, but this API is not as fully-featured as previous versions of ConceptNet due to its larger size rendering some of the previous queries intractable. ConceptNet 5 uses a different, larger set of relations than previous versions.

ConceptNet contains a distinct sort of information—common sense. Its assertions are not limited to statements that are always, unequivocally true. For instance, ConceptNet 4 contains the assertion "sport At Location field." It's not always true that sports are found on a field, asplenty of sports are played indoors. However, this assertion is still a useful bit of common sense and although it would not apply to a story about a volleyball game, it could certainly help in understanding a story about a football game. Commonsense reasoning is importantly different from logical reasoning, as logical reasoning would combine the true assertions "volleyball Is A sport" and "sport At Location field" to form the conclusion "volleyball At Location field," which is rarely true because volleyball is usually played indoors or on a beach. Commonsense

reasoning is also not statistical. The score for "sport At Location field" is not determined by dividing the number of sports played on a field by the number of sports, as such an approach clumsily ignores context dependency. Instead, ConceptNet embraces the very nature of common sense—assertions can be ambiguous, contradictory, or context-dependent, but all of those properties apply to the common sense that humans possess as well. ConceptNet's lack of precision does mean applications that use ConceptNet must be careful about how they appy its knowledge, though. Chapter 6 discusses ASPIRE and the rather involved method I developed for allowing Genesis to use goal-related ConceptNet knowledge independently of user-defined rules.

3.2 AnalogySpace

In addition to strictly human-submitted knowledge, ConceptNet 4 offers a supplementary commonsense knowledge base, AnalogySpace. The original English knowledge base is fairly sparse, with an approximate average of ten assertions about any given concept (the variation is quite high, however—most concepts have six assertions or fewer, while some of the most popular concepts have thousands). This sparsity is not because most of the contained concepts are unrelated and disparate; rather, the collected knowledge about many concepts only describes a very small fraction of these concepts' properties. Simply put, ConceptNet doesn't know a lot about many concepts because it hasn't been told a lot about many concepts.

AnalogySpace was created to address the sparsity problem and adds knowledge by cleverly applying a popular matrix algebra technique on the original knowledge base. ConceptNet can be thought of as a three-dimensional matrix of assertions where the dimensions are the left concept, the relation, and right concept. The value in each matrix entry is the score of that assertion, or zero if that assertion is not present. This matrix is very sparse, but singular value decomposition can be used to fill in many of its gaps.

Singular value decomposition (SVD) is a matrix factorization method which separates a matrix into a sum of lower-rank matrices. The terms of the sum progressively

increase in rank, so as each term is added to the sum, the sum better approximates the matrix. When the last term is added, the sum is exactly equal to the original matrix. AnalogySpace is formed by computing the SVD of the original version of ConceptNet and truncating the matrix sum so that the result is a low-rank approximation of the original matrix. This low-rank approximation ignores many of the details and idiosyncrasies of the data, instead focusing on bigger trends and relationships. This creates a transfer of information between similar concepts. For example, if the original knowledge base knows that apples and oranges have many of the same properties, but only knows that apples are used for juice, AnalogySpace has evidence that oranges are also used for juice. Therefore, although the assertion "orange Used For juice" has a score of 0 in the original matrix, the assertion has a higher score in the AnalogySpace matrix. This behavior is naturally produced by the SVD, a very powerful matrix operation (Havasi et al., 2009).

ConceptNet 4 provides separate API calls for querying the value of an AnalogySpace assertion versus traditional ConceptNet assertions. AnalogySpace also supports many other powerful queries such as calculating the similarity and relatedness of two concepts.

3.3 Discussion

ConceptNet's data is not perfect and does contain some noise. However, the ability for users to rate assertions submitted by others helped mitigate spam submissions. AnalogySpace was also used to detect suspicious data and suggest assertions for users to examine (Havasi et al., 2009; Speer, 2007).

All of ConceptNet's knowledge is binary, describing how two concepts are related. This makes it importantly different from a Genesis rule, which can have an unbounded number of antecedents and consequents. Genesis rules are more appropriate for describing domain-specific rules that typically have more nuance than just relating two concepts. ConceptNet knowledge is simpler and higher-level, but still quite valuable. These two ways of representing common sense, ConceptNet assertions and Genesis

rules, are compared at greater length in section 6.4.

ConceptNet 4 offers a broad and rich set of supported queries in its API, but ConceptNet 5 only offers a subset of these queries. Because ConceptNet 5 is significantly larger in size, many of the techniques ConceptNet 4 uses to support its queries are no longer performant in the following version, resulting in loss of support for many interesting queries. For instance, ConceptNet 4 can compute both concept relatedness and similarity using AnalogySpace. However, the approach used to generate AnalogySpace does not scale to ConceptNet 5. Instead, ConceptNet 5 computes concept relatedness using word embeddings derived from many different datasets, but does not support concept similarity because similarity is not captured by this technique (Speer & Chin, 2016). Word embeddings are discussed at greater length in section 9.1.

Similarity and relatedness are importantly different. "Desk" and "paperclip" are related because they're both found in an office, but they are not similar because they have very different uses. "Desk" and "table" are similar, though, because they're both used to support objects. I used both ConceptNet 4 similarity and ConceptNet 5 relatedness in my work, as explored further in Chapter 4. While ConceptNet 5's larger size does allow it to represent more knowledge, its lack of support for many interesting queries is a drawback.

Both ConceptNet 4 and ConceptNet 5 contain commonsense knowledge, but ConceptNet 4's content is especially valuable because every assertion was explicitly contributed by a human. While resources such as WordNet and Wikipedia undoubtedly have their uses, the knowledge contained in them cannot always be considered common sense, and their presence in ConceptNet 5's knowledge dilutes the common sense that humans possess. Moreover, ConceptNet 5 does not contain all of the assertions that ConceptNet 4 does. For example, ConceptNet 5 contains the obscure knowledge that a peripatetic is a type of pedestrian while ConceptNet 4 does not, but ConceptNet 4 knows the more obvious fact that pedestrians are found on sidewalks while ConceptNet 5 does not. Because my focus is on the use of common sense in story understanding, not encyclopedic knowledge, ConceptNet 4 fit more naturally

into my work, but each knowledge base has its strengths and weaknesses. Therefore, my implementation primarily uses ConceptNet 4, only consulting ConceptNet 5 for its relatedness information as discussed in Chapter 4.

Chapter 4

Connecting Genesis and ConceptNet

I established a general connection between Genesis and ConceptNet so wide varieties of commonsense knowledge can be incorporated into story processing. I implemented several types of ConceptNet queries within Genesis, but designed the connection so it's easily extensible to other queries with different kinds of results. I also held user-friendliness as an important design goal when constructing the link between the two systems, as I want it to be as simple as possible for Genesis programmers to incorporate ConceptNet knowledge into their programs without worrying about the messy networking details.

ConceptNet 4 offers an HTTP API in which the client submits POST requests containing query parameters to the server and receives a JSON response. I used Java's built in networking library to submit the requests and the Jackson Java library to parse the responses and turn them into Java objects (*Jackson Project*, n.d.). All of the networking calls are hidden "under the covers" in the ConceptNetClient class, which offers convenient and simple public methods enabling users to easily access and interpret ConceptNet information.

I modeled many of the ideas in ConceptNet as classes in Genesis, such as ConceptNetAssertion. These classes standardize the representation used for interaction with ConceptNet even before the query or networking stages. They also provide a convenient way to normalize the information the objects represent. For instance, the ConceptNetConcept class, used to represent a concept, "cleans up" its user-given

concept String before storing it to ensure the String is in the standardized format that the ConceptNet queries require. Because these model objects represent ideas in ConceptNet rather than query or networking details, they are visible to the rest of Genesis's code.

All types of ConceptNet queries implement the ConceptNetQuery interface, which standardizes important information about each query. Every query knows what arguments it must send in the POST request, the object type used to represent the result of the query, how to parse the server's response into that object type, and what the default result of the query should be if there's any sort of error contacting the server or if ConceptNet does not have the necessary concepts in its knowledge base. Subtypes of ConceptNetQuery include ConceptNetAssertionQuery, ConceptNetSimilarityQuery, and ConceptNetConceptExistsQuery. ConceptNetAssertionQuery and ConceptNetSimilarityQuery have a result type of Double because they receive a score from the server, while ConceptNetConceptExistsQuery has a result type of Boolean.

ConceptNetWideQuery, another subtype of ConceptNetQuery, is an abstract class which represents queries that expect a list of results rather than a single result. For instance, ConceptNetFeatureQuery extends ConceptNetWideQuery and is used to send open-ended queries about features to ConceptNet. As introduced in section 3.1, a feature in ConceptNet is an assertion with one of its concepts missing, such as "person Desires ____" or "____ Causes guilt." A ConceptNetFeatureQuery takes in a feature, asks the server for all the ways this feature could be completed given its commonsense knowledge, and returns a list of scored, complete assertions back to the user. Such a query is useful if Genesis isn't sure exactly what particular knowledge it may need in the future about, say, what people desire, so it wants to get all of ConceptNet's knowledge describing people's desires now and see what turns out to be relevant later.

Results of ConceptNet queries are packaged up into instances of the ConceptNetQueryResult class, which is visible to general Genesis code. This class contains the result of a query, such as a score Double value, and the original ConceptNetQuery that gave this result. The ConceptNetQueryResult class is useful because it can be easily

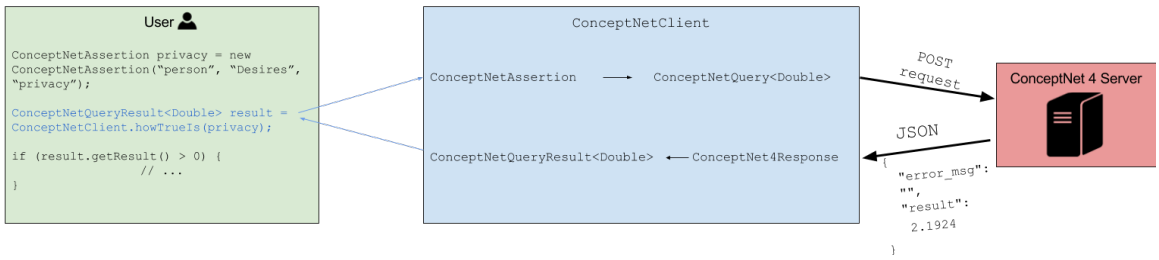

Figure 4.1: The organization of the Genesis-ConceptNet connection. The ConceptNetClient class sits as an intermediary and allows Genesis programmers to easily incorporate commonsense knowledge without worrying about any networking details. Icons courtesy of sharma and aLf at The Noun Project and are used under a CC-BY 3.0 license.

turned into a ConceptNetJustification object. These justifications provide a simple way for Genesis to explain how a given bit of ConceptNet knowledge affected its story processing. ConceptNetJustification has a `toJustificationString()` method which returns a user-friendly justification String explaining a piece of ConceptNet knowledge in terms of the query that was used to obtain the knowledge and the result of the query. These justifications are used to show the user what knowledge was received from ConceptNet while processing a story and how that knowledge was used, making Genesis's commonsense knowledge as transparent as possible.

Instances of ConceptNetQuery are never created by the external Genesis programmer. Instead, ConceptNetClient constructs these queries and sends them off to the server. For instance, ConceptNetClient offers a `howTrueIs()` method which takes in a ConceptNetAssertion given by the user, constructs a ConceptNetQuery, uses this query to send a POST request to the ConceptNet 4 server, parses the returned JSON into a ConceptNetQueryResult object, and returns this object to the user. Figure 4.1 shows an example of this data flow. Genesis programmers looking to use ConceptNet knowledge need to interact only with the instances of the classes used to model ConceptNet ideas, ConceptNetJustification instances, and ConceptNetQueryResult instances; all other classes are used for networking details and are package-protected because they are part of the internal operation of ConceptNetClient.

ConceptNetClient contains a caching layer which is invisible to the user. The

client maintains a Map between ConceptNetQuery instances and their corresponding ConceptNetQueryResult instance. When a ConceptNetQuery instance is formed in response to a user method call, the client first checks if the instance is contained in the cache. If so, the client returns the corresponding result to the user. If not, the ConceptNet server is contacted and a ConceptNetQueryResult instance is formed. Before the result is returned to the user, it is first placed into the cache so the result will be remembered. The cache is written to a file when Genesis is closed and read from the file when Genesis is opened, so it persists between Genesis sessions just as the START and WordNet caches do. The use of this cache assumes ConceptNet knowledge will not regularly change. The Genesis user interface allows the user to control whether or not the cache is being used and gives the ability to purge the cache entirely, so in the event of a ConceptNet update the user does have a way to ensure the latest knowledge is being used.

Currently, nearly all of Genesis's queries consult ConceptNet 4, not ConceptNet 5. ConceptNet 4 has a broader API than ConceptNet 5, so ConceptNet 4 is the only option to serve many of the queries ConceptNetClient offers. However, even when both versions support a certain query, using ConceptNet 4 is nearly always preferable because ConceptNet 4's knowledge better aligns with human common sense while ConceptNet 5's knowledge is more encyclopedic in nature. This distinction is discussed in more detail in section 3.3.

Both versions are used to answer queries about how similar two concepts are. The two use different methods to compute association between concepts—ConceptNet 4 computes similarity using the matrix factorization methods discussed in section 3.2, while ConceptNet 5 computes relatedness by comparing word embeddings formed from many different datasets (Havasi et al., 2009; Speer & Havasi, 2012) as discussed in sections 3.3 and 9.1. I've found that combining ConceptNet 4's similarity score with ConceptNet 5's relatedness score gives a more robust score than using either one alone, as the two complement each other's weaknesses. Specifically, if the user client asks the client for a similarity score for two concepts and the concepts are presents in both versions of ConceptNet, the client returns the geometric mean of the scores

from the two versions. If both versions do not have both concepts, a score of zero is returned. The geometric mean was chosen instead of a simpler means of combination such as the average because the geometric mean biases the result towards the lower of the two scores. This method is more generous than returning the minimum, but less generous than returning the average. This query is currently the only place that ConceptNet 5 is being used in my implementation.

The resulting connection between the two systems is layered, general, and easy-to-use. Users of ConceptNetClient do not need to concern themselves with any networking details or how the client works internally, and the ConceptNetQuery infrastructure can easily be extended to support new types of queries and query results. These attractive qualities make incorporating arbitrary ConceptNet knowledge into the existing architecture a straightforward task for future Genesis researchers.

Chapter 5

Common Sense Enabled Rule Matching

5.1 Overview

Common sense enabled rule matching (CSERM) is an immediate application of ConceptNet which, for many stories, decreases the size of the Genesis ruleset necessary for identifying causal connections. In its default mode, Genesis's rule matching is relatively limited. Consider a simple example story with a rule "If XX eats food, XX becomes full" and event "Matt eats food." Genesis matches Matt eating food to the antecedent of the rule and infers that Matt is full because he ate food. If the event were "Matt eats an apple," Genesis would also be able to infer that Matt is full because it consults WordNet when processing every entity in the story, and WordNet tells it that an apple is a type of food (along with it being an instance of produce, an edible fruit, a solid, etc.).

However, the knowledge that WordNet provides is not always comprehensive, which can lead to Genesis failing to realize that a rule in its ruleset is relevant to a story event. If such a rule fails to fire, Genesis's interpretation of the story lacks causal connections, and Genesis sees the story as less coherent than it actually is. For instance, consider a different example story with a rule "If XX harms YY, YY becomes angry" and the event "Matt attacks Josh." WordNet tells Genesis that

attacking someone is an action, but it does not specify that attacking is a type of harm.

WordNet is very good at providing accurate hierarchical descriptions of nouns because its knowledge is more encyclopedic than common sense. As discussed in section 2.1, WordNet knows that a dog is a canine and a mammal, but does not tell Genesis that dogs can be pets even though dogs are frequently pets. The hierarchies WordNet provides are especially weak with verbs. ConceptNet addresses many of these weaknesses. Because ConceptNet describes commonsense knowledge, it provides information about how these words are actually used rather than providing strict hierarchical classifications. ConceptNet can tell Genesis that "attack" is similar to "harm" so that if there are rules that discuss harming, story events that contain attacking can still trigger these rules. In this way, Genesis can more effectively employ its ruleset using common sense because a rule applies to not only story events that exactly match the rule's antecedents, but also story events that are similar to the rule's antecedents. This means the user does not have to construct nearly as many rules, and the rules she does create can discuss a general concept such as "harm" and still apply to any specific sort of harm that may be contained in the story such as "attack." With CSERM, the applicability of a rule can amplify by a factor of approximately ten to one hundred.

5.2 Implementation

CSERM is an extension of Genesis's normal rule matching which uses WordNet. The job of CSERM is to determine if two Genesis entities, one of which is a rule, are similar enough that the rule should fire. CSERM can be enabled or disabled by a switch in the Genesis user interface. This feature currently only compares the verbs in two relations for similarity. Recall that a Genesis entity is an inner language representation of a sentence fragment or sentence, and a relation is a type of entity that describes a connection between two entities (see section 2.1 for more information). For example, "Matt attacks Josh" is a relation with "attack" as its type, "Matt" as its subject, and

"Josh" as its object. I implemented CSERM specifically for relations because rules most commonly describe connections between relations. However, CSERM could be extended to support other types of entities if desired.

CSERM starts out by invoking traditional matching between the rule antecedent and the story event entity. If traditional matching succeeds, CSERM returns this result because there's no need to incorporate ConceptNet knowledge. Otherwise, the next step is to form a structure mapping between the rule antecedent and the story event. Consider a rule "If XX harms YY, then YY become angry" and a story event "Matt attacks Josh." The rule antecedent here is "XX harms YY." Below is a textual representation of the inner language Genesis uses to represent this rule antecedent and story event, respectively:

```
(rel harm (ent xx-108) (seq roles (fun object (ent yy-100))))
(rel attack (ent matt-210) (seq roles (fun object (ent josh-216))))
```

"Rel" refers to a relation, "ent" to an entity, "seq" to a sequence. "Roles", "fun," and "object" are used to describe the role the later objects play in the sentence (direct objects). The numbers following each of the entities XX, YY, Matt, and Josh are used to represent specific instances of objects with this name. Structure mapping attempts to form a set of bindings between these two entities based on their structure. Structure mapping only focuses on the structure of the two entities, ignoring details such as the relation type or names of the people involved. Here, the structure mapping would produce the bindings Matt$\longleftrightarrow$XX and Josh$\longleftrightarrow$YY because these pairs of entities correspond with each other in the overall identical structure of the two relations. If the two relations did not have identical structure, an empty set of bindings would be returned. Because there can be multiple possible structure mappings between two given entities, a list of all structure mappings is calculated rather than just one.

If the structure mapping fails, CSERM concludes there's no match and returns. Otherwise, the list of structure mappings is filtered down to just the ones that are valid. Validity here means every entity is bound to exactly one other entity and the bindings agree with any bindings that have been accumulated from matching

previous antecedents in the same rule. Validity also requires that no binding maps a person to a non-person and that all pairs of non-people entities that are bound together match using traditional Genesis matching. We place these restrictions on the structure mapping to ensure that the story event and rule antecedent not only have identical structure, but also have logically corresponding components.

After filtering down the possible structure mappings as just described, the ConceptNetClient class is used to contact ConceptNet and get a similarity score for the types of each pair of relations (i.e. the relation verbs) in every structure mapping. As discussed in Chapter 4, the score returned from ConceptNetClient is actually a function of ConceptNet 4's similarity score and ConceptNet 5's relatedness score. If there is a structure mapping in the valid list that has relation pairs with similarity scores all greater than or equal to a cutoff value (0.33), the two relations are considered similar and the rule match succeeds. If no structure mapping has relation pairs that are all similar, the rule match fails.

Once a rule match has been discovered, the procedure returns the associated bindings to signal that a successful match has been found. The procedure also attaches a ConceptNet justification property to these bindings to explain why this match was made (properties are discussed more in section 2.1). Because this match goes beyond using explicit Genesis rules or WordNet data and incorporates the subtler ConceptNet similarity information, it's useful to capture a justification for the match so it can be displayed to the user and explain why Genesis made this causal connection. The ConceptNetQueryResult obtained from sending a similarity query to ConceptNet is converted into a ConceptNetJustification, and the rule that CSERM used as it base is added to this justification. This justification object is added as the value of the ConceptNet justification property on the returned bindings. When the causal connection is finally made that resulted from this match, a Genesis entity that represents the causal connection is instantiated. All of the accumulated properties on the bindings are added as properties on the causal connection entity.

5.3 Discussion

CSERM has two important limitations: it can introduce a significant slowdown on stories with large rulesets, and it relies on a cutoff value to determine how high a similarity score must be for two concepts to be treated as similar. Section 8.1 discusses the potential slowdown and a possible fix. Currently, the cutoff is 0.33, an experimentally chosen value. Section 8.2 discusses a method for making this cutoff value dynamically adjust in response to story processing rather than a static, carefully chosen constant.

I have observed CSERM to be quite powerful. The technique can amplify the applicability of a rule by a factor of ten to one hundred. Of course, this amplification factor is not uniform and depends on the chosen cutoff and the generality of the verbs within the rule. Section 7.4 discusses observed rule amplification factors and their derivations.

Genesis uses its "dereferencing" process to resolve equivalent entities or parts of entities into the same object. All entities added to the story are fed through this dereferencing process before they are actually added. However, dereferencing can result in entity properties getting lost if care is not taken to preserve them. I developed the notion of identifier properties to preserve entity properties. Identifier properties are properties intrinsic to the entity's identity, meaning that an entity with an identifier property and an otherwise equivalent entity without that property will not be resolved into the same object during story processing. Any property can be set as an identifier property so that it will be used in this resolution process. The ConceptNet justification property is used as an identifier property.

The similarity information ConceptNet provides is useful in a different way than a resource such as WordNet or a thesaurus. WordNet is constrained by its hierarchical structure, and while a thesaurus does provide similar words, this similarity is very formal and doesn't usually capture much commonsense knowledge about how these words are actually used. For instance, ConceptNet knows that words such as "deceive," "threaten," "rob," "anger," "attack," "torture," "dishonor," "murder," and

"embarrass" are all similar to "harm," and a thesaurus does not. ConceptNet's vast amount of commonsense knowledge influences what it sees as similar, resulting in a unique but useful variety of similarity. Other resources have their strengths, but I believe ConceptNet's notion of similarity to be special.

CSERM is an effective way of expanding Genesis's ruleset by applying its rules in a commonsense manner. Previously, a story's events had to nearly exactly match rule antecedents in order for the rule to fire. Now, Genesis can apply its rules much more flexibly by identifying when story events are similar enough to a rule's antecedents that the rule should fire. This allows the user to write more general rules that cover a wide variety of commonsense applications instead of specifying each specific application in its own rule. CSERM is an important step in cutting down the number of user-defined rules necessary for story understanding, but it still requires the user to construct rules. Chapter 6 examines a way that Genesis can use ConceptNet's knowledge instead of Genesis rules to make causal connections, allowing for effective story understanding with drastically fewer or even zero user-given rules.

Chapter 6

ASPIRE

6.1 Overview

Goals are essential to human nature. Humans are constantly forming goals and taking actions towards them. Kenrick, Neuberg, Griskevicius, Becker, and Schaller articulated the fundamental-motives framework which describes how instinctual human goals such as self-preservation, finding a mate, and child rearing affect our biology, cognition, and behavior. These low-level goals can change the way the brain processes information, altering what people pay attention to and what people remember (Kenrick, Neuberg, Griskevicius, Becker, & Schaller, 2010). At a more conscious level, goals are an effective motivational tool. Task performance improves when specific, measurable goals are set and progress towards these goals is tracked (Locke & Latham, 2006). Because goals drive humans at both a subconscious and conscious level, goals are prevalent in the stories humans tell as well. Schank recognized the importance of identifying character goals when processing natural language in 1977. Enabling Genesis to better understand goals, their causes, and actions taken towards completing them improves Genesis's comprehension of goal-oriented stories. Furthermore, getting Genesis to understand these stories with a minimal set of user-defined rules is a big step towards allowing Genesis to read and understand naturally occurring stories with little user effort.

ConceptNet is a great resource for analyzing goals because it contains a multitude

of commonsense knowledge about what causes goals and what fulfills goals. The relations "Causes Desire," "Motivated By Goal," "Used For," and "Has Subevent" are all particularly relevant to the goal domain. ConceptNet 4 contains knowledge such as reading is motivated by the goal of learning and needing money causes the desire to apply for a job. I have enabled Genesis to leverage this knowledge while processing a story to both identify events that cause characters to have a certain goal and keep track of the actions they take to complete these goals. By using the knowledge contained in ConceptNet, the ruleset necessary for understanding goal-centric stories can be quite minimal. Depending on the content of the story and how well it aligns with ConceptNet's knowledge, the need for a user-defined ruleset can even be entirely eliminated!

I established ASPIRE's framework of analyzing characters' goals, their causes, and the actions taken to complete them, and I implemented the ASPIRE system in approximately 650 lines. ASPIRE works closely with ConceptNet, but is not dependent on it; it is able to function using Genesis rules and WordNet knowledge as well. However, ConceptNet makes ASPIRE much more powerful because it allows ASPIRE to function independently from user-defined rules; instead, ASPIRE uses common sense from ConceptNet, and the ConceptNet knowledge currently incorporated is equivalent to approximately 30,000 rules. With ASPIRE and ConceptNet, Genesis can much more completely identify the causal relationships between the story's events with far less user effort, and all of Genesis's powerful story analysis capabilities build off the understanding ASPIRE grants Genesis. Without ASPIRE and ConceptNet, Genesis would fail to see how many, if not all, of a goal-centric story's events are connected, and any further story analysis would greatly suffer.

6.2 Framework

More specifically, ASPIRE works by maintaining a list of candidate character goals as Genesis sequentially reads the story, adding to the list when it detects that a story event might cause a character in the story to have a goal. ASPIRE analyzes

every received event to see if it causes a candidate character goal or contributes to any of the existing ones. It even applies this analysis to inferred events that ASPIRE itself created, allowing its inferences to build off one another. For instance, consider a story about a man named Sean that contains the event "Sean is gaining weight." ASPIRE checks if this event causes a goal by looking at the rules Genesis was given and by consulting ConceptNet. To query ConceptNet, ASPIRE forms the feature "gain weight Causes Desire ____" and uses ConceptNetClient to get a list of scored ConceptNet assertions that complete this feature. In this case, ConceptNet knows that gaining weight causes the desire of exercising, so the returned list contains the assertion "gain weight Causes Desire exercise" among other relevant assertions. Therefore, ASPIRE adds the candidate character goal "Sean wants to exercise" to its list and continues reading the story.

Suppose the story continues and contains some later event "Sean rides his bike to work." When it receives this event (and for every received event), ASPIRE iterates over its candidate character goal list and checks if the event contributes to each candidate character goal. To detect such a contribution, ASPIRE first tries to match the event to the candidate character goal using traditional Genesis matching. This would take care of events such as "Sean exercises," which exactly matches the exercise candidate character goal, and "Sean works out," which matches the exercise candidate character goal with the help of WordNet because "work out" fits cleanly under "exercise" in WordNet's hierarchy. If this match succeeds, then ASPIRE concludes that the event contributes to the candidate character goal. Otherwise, ConceptNet is consulted because traditional matching failed. When processing the event "Sean rides his bike to work," ASPIRE extracts the concept "ride bike" from the event, "exercise" from the candidate character goal, and uses ConceptNetClient to obtain the scores for the assertions "ride bike Motivated By Goal exercise," "ride bike Used For exercise," and "exercise Has Subevent ride bike." If ConceptNet confirms any of these assertions, ASPIRE concludes that the event contributes to the candidate character goal; otherwise, with both traditional matching and ConceptNet failing to make any connection between the event and the goal, ASPIRE concludes the event does not

contribute to the goal.

If ASPIRE concludes that a story event contributes to a candidate character goal, Genesis forms causal connections and adds new inferred events to the story based on ASPIRE's analysis. Genesis instantiates the candidate character goal and explicitly adds it to the story because ASPIRE has inferred that this goal was present. Returning to the Sean example, Genesis adds "Sean wants to exercise" to the story. Genesis also forms causal connections between the inferred goal's cause, the inferred goal, and the action taken that contributes to the inferred goal (i.e. "Sean gains weight," "Sean wants to exercise," and "Sean rides his bike to work," respectively). Genesis goes one step further by instantiating the completion of the goal and adding it to the story as well ("Sean exercises"). Genesis explicitly adds the inferred goal and inferred goal completion to the story so that these inferences can interact with other Genesis rules that discuss the desire to exercise or exercise itself. Without explicitly acknowledging that Sean exercises when he rides his bike, Genesis rules about exercise would not fire because Genesis does not match "ride bike" with "exercise." Furthermore, this explicit acknowledgment allows "Sean exercises" to feed back into ASPIRE and cause or contribute to other goals. For instance, Sean exercising could contribute to a candidate character goal "Sean wants to lose weight" which was formed earlier in the story. By explicitly instantiating both the inferred goal and the inferred goal completion and adding them to the story, ASPIRE makes plain the common sense that's obvious to humans so that it can process these commonsense inferences just as it processes explicit story events. Genesis shows the casual connections connecting story events in its cause graph display, and the cause graph for the exercise example is shown in Figure 6.1

Note that the inferred goal and inferred goal completion are only inserted into the story once ASPIRE sees an action taken that contributes to a candidate character goal. Candidate character goals on their own do not affect the story in any way—a character action must first prove a candidate goal credible before Genesis acts on ASPIRE's analysis. In this way, ASPIRE is enabling Genesis to infer the middle element, a character goal, in a latent causal chain connecting the cause of the goal,

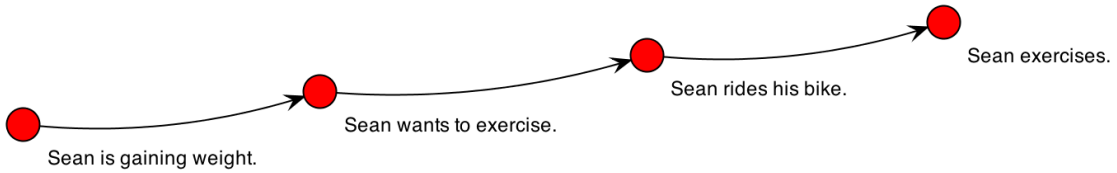

Figure 6.1: An example of a cause graph produced by ASPIRE. Cause graphs show how different story events are causally related. Story events are represented with circles and arrows signify a causal relationship. Here, ASPIRE has made two inferences, "Sean wants to exercise" and "Sean exercises," from two explicitly stated story events, "Sean is gaining weight" and "Sean rides his bike to work."

the goal, and an action taken towards the goal. If this chain is not complete, the inferred goal remains a candidate—a potential interpretation of the story's events, but one that lacks sufficient evidence. It's possible that Sean's weight gain caused him to diet instead of exercise, or maybe he didn't take any response at all. Therefore, the candidate character goal "Sean wants to exercise" is just an idea Genesis is holding in the back of its mind until it sees evidence supporting it.

The way ASPIRE analyzes goals is reminiscent of Marvin Minsky's IF-DO-THEN rules proposed in his book Society of Mind (Minsky, 1988). If Sean is gaining weight, he may want to exercise (DO); then, he rides his bike. In this chain, ASPIRE is inferring the DO from the IF and the THEN. We can shift this IF-DO-THEN chain over to apply to the goal completion, as well: If Sean wants to exercise, he may take the action of riding his bike (DO); then, he has exercised. Given the middle element in this chain (the action) and evidence that the IF is true (the cause of the candidate character goal), Genesis instantiates the IF (the candidate character goal) and the THEN (the goal completion).

6.3 Implementation

ASPIRE only requires processing one cause of a potential goal to form a candidate character goal. Similarly, ASPIRE only requires one goal contribution action to add the inferred character goal and goal completion to the story. While it is usually safe to add the inferred goal after seeing one potential cause, it's far less safe to add the

inferred goal completion after only seeing one action because ASPIRE lacks the ability to distinguish actions that complete a goal from actions that merely contribute to a goal. However, it's not immediately clear how ASPIRE would acquire such knowledge, as ConceptNet does not provide it. This limitation is discussed further in section 8.1.

ASPIRE uses ConceptNetClient to query ConceptNet for information about goals, specifically using the "Causes Desire" relation to infer goal causation and the "Motivated By Goal," "Used For," and "Has Subevent" relations to infer an action's contribution to a goal. ConceptNetClient contacts ConceptNet 4 to gather this information as discussed in Chapter 4. ASPIRE repeatedly queries ConceptNet over the course of reading a story, receiving scores for assertions of interest. However, to decide whether an assertion is true and should be acted upon, ASPIRE compares the received score to a cutoff value. If the score is greater than or equal to the cutoff value, ASPIRE interprets the assertion as true; otherwise, the assertion is seen as false and does not affect story processing. Currently, the cutoff is a constant (0.3), just as the common sense enabled rule matching cutoff is fixed. However, section 8.2 discusses ways this value can be dynamically chosen.

ASPIRE detects both candidate character goals and goal contribution actions for every story event it receives. Events are not constrained to relate to just one goal—an event can cause multiple candidate character goals, and an event can contribute to multiple goals.

6.3.1 Interaction with Genesis Rules

In addition to ConceptNet knowledge, ASPIRE uses Genesis rules to detect when a story event results in a candidate character goal. When first initialized, ASPIRE reads over all the Genesis rules, checking if each rule's consequent describes a character getting a goal (e.g. "If XX loves YY, XX may want to kiss YY"). If it does, ASPIRE stores the goal and its cause. When ASPIRE receives a story event and decides if this event causes any goals, it scans over this list of goals and their causes in addition to asking ConceptNet what it thinks. If the story event matches any of the stored goal causes, that goal is added as a candidate character goal. ASPIRE does not currently

support Genesis rules that describe goal contribution actions (e.g. "If XX wants to kiss YY, XX may embrace YY"), but the system could easily be extended to have this capability.

When a story event matches a Genesis rule that results in a goal, a candidate character goal is formed which substitutes the bindings from the match into the goal. For example, if ASPIRE received the rule "If XX loves YY, XX may want to kiss YY" and the story event "Matt loves Jackie," the formed candidate character goal would be "Matt wants to kiss Jackie." ASPIRE performs this substitution to keep track of what character has this goal and what other people are involved. ASPIRE shouldn't match a potential later story event "Matt kisses Helen" to his goal of kissing Jackie because the people do not match! These bindings are not required to be fully specified. Suppose ASPIRE was given the rule "If XX is angry, XX may want to harm YY" and the story event "Matt is angry." Because YY has not been constrained to any particular person, ASPIRE forms the candidate character goal "Matt wants to harm someone," which can match any story event that describes Matt harming a person such as "Matt harms Josh" or "Matt harms Jackie."

6.3.2 Concept Extraction

Extracting concepts from Genesis entities requires some care. As mentioned in section 3.1, although user-defined story patterns such as those describing "Pyrrhic victory" are commonly referred to as "concept"s in several other Genesis texts, here (and throughout this thesis) I am using "concept" only in the ConceptNet sense of the word. ASPIRE extracts concepts from story events when it queries ConceptNet to see what candidate character goals a story event may cause. It also extracts concepts from both the candidate character goal and the story event when determining if a story event contributes to a goal. I devised my method of concept extraction by studying the format of ConceptNet's goal-related knowledge, observing common patterns, and constructing procedures that take advantage of these patterns in a somewhat cautious yet effective manner.

Nearly all of the concepts ASPIRE extracts from Genesis entities are verbs or

verb phrases rather than nouns or noun phrases. Verbs and verb phrases are simpler to extract because their meaning is less dependent on the rest of the sentence. For instance, it's safe to extract the concept "drink beer" from "Matt drinks a beer on a hot day." However, although ConceptNet has useful goal knowledge about nouns, it's not always safe to extract a noun direct object as a concept. If ASPIRE sees that "Matt buys a book," it's appropriate to extract both "buy book" and "book" to see what goals Matt may have as a result, and ConceptNet would give ASPIRE assertions such as "book Causes Desire read." However, if the event were instead "Matt loses his book," while it would still be appropriate to extract "lose book" as a concept, "book" by itself would not be appropriate because ConceptNet provides knowledge about possessing a book, not losing it. I have not found a simple, comprehensive, computational way of deducing whether or not it's safe to extract the noun direct object as a concept, so ASPIRE takes a cautious approach and only extracts verbs or verb phrases. There are two exceptions where the noun direct object is indeed extracted: if the verb is a form of "feel" (e.g. "sad" would be extracted from "Matt feels sad") or of "have" (e.g. "compassion" would be extracted from "Matt has compassion").

Goal Causation

When ASPIRE uses ConceptNet to detect what goals a story event may cause, it can extract both verbs and verb phrases as concepts. Proper nouns never appear in extracted verb phrase concepts. ConceptNet doesn't contain knowledge about specific people, places, or things, just general knowledge, so it's not useful to form concept phrases that refer to specific story proper nouns. However, ASPIRE can still form goals related to specific characters using ConceptNet knowledge. Consider the story event "Matt loves Jackie." ASPIRE detects this event has a proper noun as its direct object, so it extracts only the concept "love" and queries ConceptNet to see what desires it causes. ConceptNet responds with assertions such as "love Causes Desire kiss," and ASPIRE forms the candidate character goal "Matt wants to kiss Jackie." It's important for the direct object, Jackie, to transfer from the cause of the goal to the goal itself. We wouldn't want Matt's love for Jackie to form the general goal

"Matt wants to kiss," as this would match unrelated story events such as "Matt kisses Helen." If a proper noun appears as the direct object in the cause of a goal, ASPIRE assumes it is also a direct object in the goal itself. This follows the observed structure of ConceptNet's knowledge. In this proper noun case, ASPIRE extracts just the verb or the verb paired with any modifying adverbs as concepts.

Concept extraction for direct objects that are not proper nouns is handled differently. In this case, only direct object verb phrases, not sole verbs, are extracted as concepts. For example, ASPIRE would extract the concept "win game" from the story event "Matt wins the game," and ConceptNet would give it knowledge such as "win game Causes Desire celebrate." Common sense tells us in this situation Matt may want to celebrate, not celebrate the game specifically, so ASPIRE does not transfer the direct object "game" to the goal in this case. While there are rare cases when transferring the direct object would be appropriate, ConceptNet 4's knowledge is almost always structured such that when the cause is a verb phrase, the goal concept in Causes Desire assertions explicitly includes any objects it takes. Therefore, there's no need for any transfer of direct object. Sole verbs are not extracted as concepts in this case because the knowledge about just the verb may not be relevant when the object is taken into account (e.g. "fight" vs. "fight inflation"), and ASPIRE strives to be as specific and precise as possible.

If the story event is intransitive, the verb by itself is extracted. Verb phrases consisting of the verb and any modifying adverbs are extracted as well.

Goal Contribution

Verbs, verb phrases, proper nouns, and transitivity also come into play during concept extraction for goal contribution detection. Here, ASPIRE must decide if a story event contributes to a candidate character goal. This task is somewhat more involved than goal causation concept extraction because concepts must be extracted from both the candidate character goal and the story event and matched together. There are three possible cases: both the story event and the goal are transitive, their transitivities do not match, or they are both intransitive.

Importantly, when matching a candidate character goal to a story event, the subjects of the two entities must match to ensure the story event describes the character of interest performing an action, not someone else. If the story event matches the candidate character goal using traditional Genesis matching, the event is identified as contributing to the goal and no concept extraction or ConceptNet communication is necessary. This traditional matching uses WordNet knowledge, so the goal "Matt wants to play a sport" would match the action "Matt plays hockey." However, traditional Genesis matching is usually not sufficient, so ASPIRE must proceed with concept extraction.

Verb phrases are used whenever one of the entities is transitive, as ASPIRE prefers to be as precise as possible and match an entire phrase instead of just the verb. As mentioned in the previous section on goal causation concept extraction, verb concept phrases do not include any proper nouns because ConceptNet does not contain information about specific people, places, or things. Instead, verb phrases contain the verb with either a modifying adverb or a direct object that isn't a proper noun.

When a story event or candidate character goal contain multiple possible concepts, ASPIRE continues extracting concepts from both and comparing the two until it detects goal contribution or it exhausts all possible concept pairings. ASPIRE currently does not continue querying ConceptNet after a match has been detected to find all possible ways that the action contributes to the goal, a decision made to improve performance. Such behavior could be easily implemented if desired.

Consider the case where both the story event and candidate character goal are transitive. ASPIRE first tries to extract direct object verb phrases from the event and goal and match them together. For instance, for the event "Matt plays violin" and candidate goal "Matt wants to make music," ASPIRE extracts "play violin" and "make music" and ConceptNet tells it "play violin Motivated By Goal make music." If ASPIRE fails to find a match, it checks if both the event and goal have the same proper noun as an object. If so, ASPIRE extracts just the verbs from the event and goal and query ConceptNet to detect any causal relationship. This handles cases such as matching the event "Matt punches Josh" to the goal "Matt wants to hurt Josh"

using the knowledge "punch Motivated By Goal hurt." If both the verb phrase and proper noun phases of matching fail in this doubly transitive case, ASPIRE concludes the event does not contribute to the goal.

Next, let us examine the case where the transitivity of the story event does not match the transitivity of the goal. It's possible for an event to not match a goal's transitivity but still contribute to the goal, such as the contribution of the event "Matt sings" to the goal "Matt wants to make music." ASPIRE first tries to match direct object verb phrases of the transitive entity against the verb and any adverbial verb phrases from the intransitive entity. If this fails, ASPIRE concludes there is no goal contribution present. ASPIRE does not extract just the verb from the transitive entity because of its preference for specificity during concept extraction.

In the third and final case, both the story event and candidate character goal are intransitive. ASPIRE extracts just the verb and any adverbial verb phrases from both entities and queries ConceptNet to see if the two are related. For example, the story event "Matt lies down" would match "Matt wants to relax" through this final case because both "lie down" and "relax" are intransitive.

Bindings

ASPIRE makes an important assumption about the structure of the ConceptNet knowledge used to form a candidate character goal or detect a goal contribution action. Throughout its processing, ASPIRE assumes at times that the bindings, or the subject and optional object between which a concept verb is nested, are consistent between the two concepts in an assertion. ASPIRE makes this assumption because it almost always matches the structure of ConceptNet knowledge, but this assumption does not invariably hold. ConceptNet unfortunately does not provide information about how the subject and objects a concept verb takes relate to the subject and objects of the other concept in an assertion, whereas Genesis rules make these bindings explicit. For example, ConceptNet 4 contains the assertion "lie Causes Desire discover truth." Common sense tells us that if Matt lies to Josh, Josh will want to discover the truth. However, given the event "Matt lies to Josh," ASPIRE would use this assertion

to form the candidate character goal "Matt wants to discover the truth" because it uses the same subject for the goal as was present in the goal cause. This limitation extends beyond goal causation—the assertion "hurt Has Subevent feel pain" also has two concepts with inconsistent bindings, and "Has Subevent" assertions are used to infer goal contribution.

Fortunately, the concepts in the vast majority of ConceptNet assertions have consistent bindings, so such an ASPIRE misunderstanding is a rarity. Furthermore, ASPIRE would only make inferences based on this misunderstanding if it sees supporting evidence in the story. For the faulty goal "Matt wants to discover the truth" to be inferred and inserted into the story, ASPIRE would need to see Matt taking actions to discover the truth. Because Matt was the one that lied, it's not likely for a story to contain events that show Matt searching for the truth. Even if ASPIRE misapplies ConceptNet knowledge to form a nonsensical idea, it needs to see explicit evidence of this idea in the story to complete an inference. Standard stories do not provide ASPIRE with any evidence of these nonsensical ideas, so it's unlikely for ASPIRE's misunderstandings to ever affect the story.

6.3.3 ASPIRE-created Properties

Whenever ASPIRE instantiates an inferred goal, inferred goal completion, or an inferred causal connection and adds this created Genesis entity to the story, it adds a property to the entity marking it as created by the ASPIRE system. Entity properties are discussed more in section 2.1. If ConceptNet knowledge was necessary in making an instantiated causal connection entity, ASPIRE also adds a ConceptNet justification property to the entity with the relevant justification object as the value of this property. These properties are used to keep track of what ASPIRE inferred and why, useful information to display to the user and necessary for presenting the elaboration graph, a central Genesis display. These properties are all used as identifier properties so that they are not lost during Genesis's dereferencing process. Identifier properties are further discussed in section 5.3.

ASPIRE adds inferred entities to the story as it's reading. However, if an inference

is later explicitly stated in the story text, Genesis must erase the ASPIRE property and its optional ConceptNet justification from this entity because it has been explicitly confirmed and is no longer an inference. This is similar to when a human reader makes a prediction or connection in their head which is later explicitly confirmed.

6.4 Recommendations for Knowledge Organization

ASPIRE is exciting because it is the first time ConceptNet knowledge has been directly incorporated into Genesis story processing completely independently from user-defined rules. This milestone introduces a discussion about what sort of knowledge belongs in ConceptNet and what sort of knowledge is better expressed as a Genesis rule. While the two representations do have some overlap, there are important differences, and even when either representation is appropriate I believe certain style guidlines should be followed for future Genesis programmers and users to maintain useful organizational structure.

ConceptNet knowledge is more limited than Genesis rules. Section 3.3 discusses how ConceptNet only describes binary relationships, while Genesis rules can contain and unbounded number of consequents and antecedents. Section 6.3.2 notes that ConceptNet knowledge does not express the subjects and objects concepts take as unambiguously as a Genesis rule, which can result in some confusion when applying a piece of ConceptNet knowledge. Additionally, concepts in ConceptNet are not disambiguated—the same concept "bark" is used for both the sound a dog makes and the part of a tree. Therefore, if the knowledge to be expressed is vulnerable to any of these ConceptNet limitations, I recommend encoding the knowledge as a Genesis rule.

Moving beyond practical limitations, I also recommend following a small set of organizational principles. ConceptNet knowledge should be common sense. While this term lacks a precise definition, it should be widely known, rather general, and feel obvious. Beyond that, I don't think many other restrictions should be applied to ConceptNet additions because it's a general resource used for projects other than

Genesis. Genesis rules are better for capturing domain-specific knowledge and more complex ideas, and their representation has more nuance to allow for these use cases. There is not a sharp distinction between knowledge better suited for ConceptNet and knowledge better suited for Genesis, but I believe following these organizational principles allows for cleaner, more consistent data in each format.

Chapter 7

Results

In this chapter I describe the performance of common sense enabled rule matching (CSERM) and the ASPIRE system on three example stories of increasing complexity. These new capabilities allow Genesis to make connections it previously could not identify; moreover, Genesis justifies these connections with human-readable text that describes the relevant knowledge obtained from ConceptNet. Although I present only three examples here, ConceptNet has a vast amount of commonsense knowledge, and CSERM and ASPIRE assist Genesis with large amounts of this knowledge whenever it's relevant in a story. This chapter also contains a derivation of approximate measures of the amount of common sense Genesis gains through interacting with ConceptNet in the ways I have implemented. Finally, I show an example of how my work allows other Genesis features to perform better. The full stories, rulesets, and ConceptNet justifications for the three example stories in this chapter are included in the Appendix.

7.1 Common Sense Enabled Rule Matching

Consider the following simple story about an instance of high-school bullying:

> Matt and Josh are persons. Matt and Josh are in high school. Matt is insecure because Matt has low self-esteem. Matt bullies Josh. Josh

becomes angry. Josh makes a plan because Josh loathes Matt. Josh embarrasses Matt. Matt is sad. Matt stops bothering Josh.

A human reader is likely to see that Josh becomes angry due to Matt's bullying, Josh embarrasses Matt as an attempt to get back at him, and this retaliation causes Matt to stop bullying Josh. A user of Genesis might construct the following ruleset to allow Genesis to make those connections:

- XX and YY are persons.
- XX may bully YY because XX is insecure.
- If XX annoys YY, XX angers YY.
- If XX angers YY, YY may hate XX.
- If XX dislikes YY, XX may harm YY.
- If XX harms YY, YY may stop harming XX.

However, this ruleset will not help Genesis very much in analyzing this story because its language differs from that of the story. The ruleset discusses general concepts such as "annoy," "hate," and "harm," whereas the story never uses these terms explicitly. Instead, the story discusses concepts such as "bully," "loathe," "embarrass," and "bother." WordNet helps Genesis in matching some of these concepts, but the information it provides is not complete enough to allow Genesis to correctly apply all relevant rules to the story's events. WordNet provides mostly hierarchical, strictly factual information, whereas many of the desired matches in this story are less precise and require common sense. WordNet is especially unhelpful in providing information about verbs, which are harder to categorize in a hierarchical manner. CSERM helps bridge this gap.

Two versions of the elaboration graph Genesis generated from reading this story are shown in Figure 7.1. In the top version, CSERM is turned off; in the bottom, turned on. Each event in the story, including some that were not explicitly stated but Genesis inferred and added to the story, is shown in a box. A line between two boxed events represents a causal connection that Genesis has established between the two events. Connections made with the help of ConceptNet are dotted. The elaboration graph uses an extensive color scheme to precisely specify the type of causal connection or inferred entity which I omit because it is not directly relevant

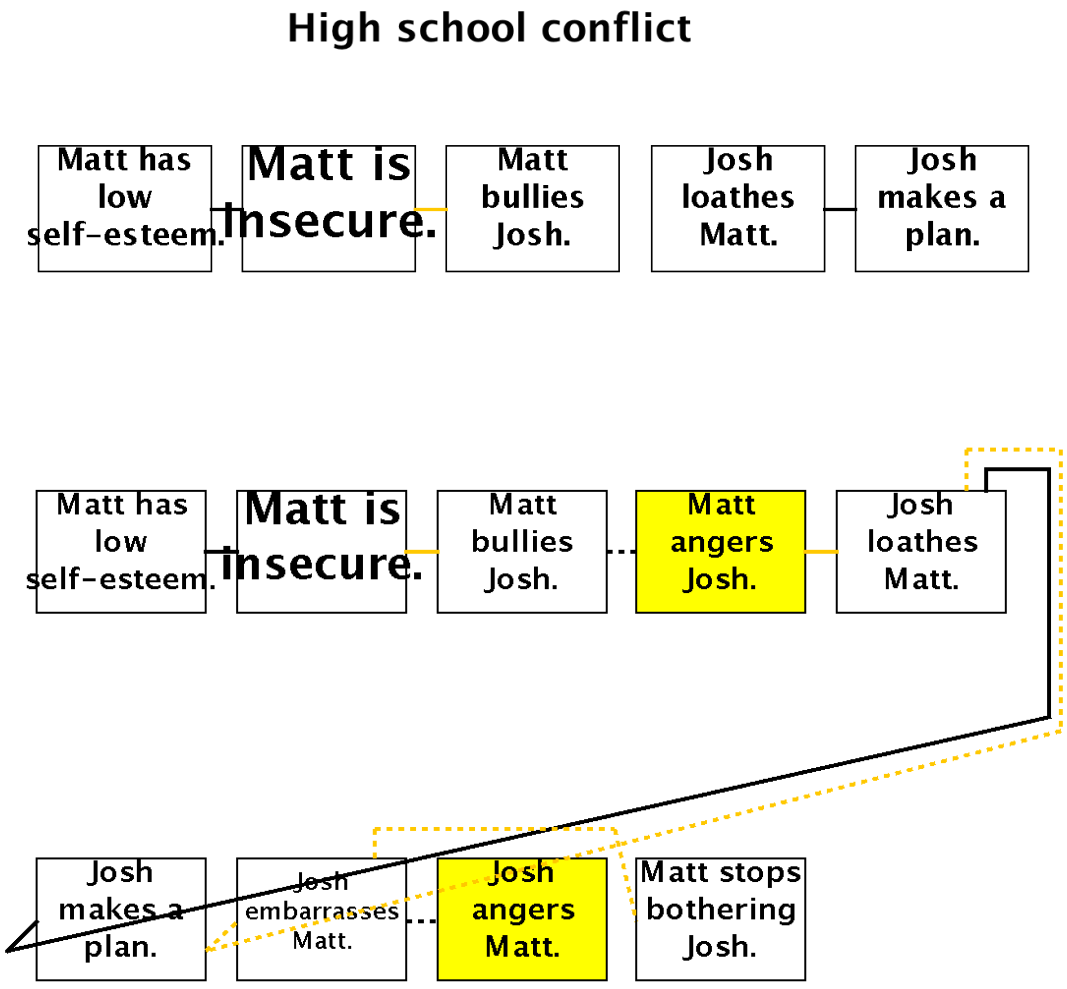

Figure 7.1: Two elaboration graphs generated by Genesis for the bullying story. In the top graph, CSERM is turned off; in the bottom, turned on. Connections made with the help of ConceptNet are dotted. CSERM doubles the number of events in the elaboration graph and increases the number of causal connections formed from three to ten.

my work; regardless of color, all dotted connections indicate ConceptNet knowledge was necessary in forming the connection.

CSERM nearly doubles the number of events in the elaboration graph because Genesis is now able to apply the rules to the story more completely. The number of casual connections identified increases from three to eight. With CSERM, Genesis is able to connect Matt's initial bullying of Josh with Josh's loathing of Matt, the loathing with Josh striking back, and the retaliation with Matt ending his bullying activity. Because these additional causal connections were made, the rest of Genesis's features can thrive off the more complete reasoning substrate. For instance, summarization and question answering depend heavily on the identified causal connections, so CSERM has given Genesis the ability to better complete other reasoning tasks related to this story.

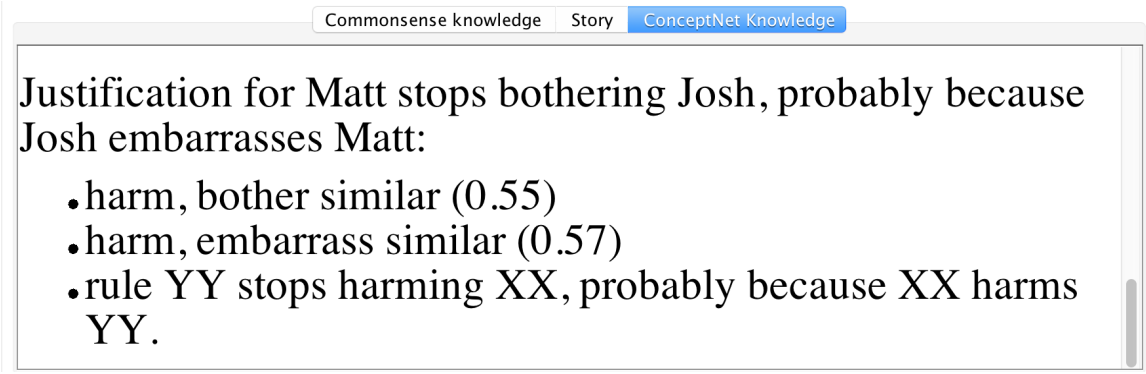

Figure 7.2: A justification produced by Genesis explaining why a common sense enabled rule match was made. Each justification names the causal connection, the rule used as a base for CSERM, and the relevant scored similarity knowledge.

Genesis is also able to explain why and how these additional connections were made. Figure 7.2 shows a section of the "ConceptNet Knowledge" pane of the Sources section which explains why Genesis believes Josh embarrassing Matt causes Matt to stop bothering Josh. The provided justification states that harm and bother are similar and harm and embarrass are similar, gives the ConceptNet scores for their similarity, and references the rule that is the basis of the similarity match. Note that these similarity scores are combinations of the ConceptNet 4 and ConceptNet 5 scores as described in Chapter 4. WordNet threads are used for a couple parts

of this story analysis, such as enabling "loathe" in the story text to match "dislike" in the story rules. These WordNet matches are typically straightforward. All of the ConceptNet matches provide justification in this panel, though, because these matches are typically more involved. This "ConceptNet Knowledge" pane is displayed alongside the "Commonsense Knowledge" pane, which holds the rules Genesis was given, and the "Story" pane, which holds the explicit story text.

7.2 ASPIRE

The following is a relatable, simple story titled "A Long Day" about a tired worker and his path towards progress:

> Fred is a person. John is a person. John is Fred's roommate. Fred works hard at the office. Fred leaves work. Fred buys beer. Fred comes home. Fred drinks beer and watches television. Fred lies down. John comes home. John plays loud music and wakes up Fred. Fred sighs from exasperation. Fred leaves the apartment and takes a walk. The quiet night helps Fred clear his mind. Fred calms down and heads home.

Although there are many possible interpretations of the story, a human reader may see the actions Fred takes after work as part of an effort to wind down after a hard day's work. It's also fairly natural to see that when Fred is disturbed by his roommate, he gets upset and takes a walk as part of an effort to calm down. A Genesis user might give it the following rule to help it better understand the story's events:

- If XX is resting and YY disturbs XX, then XX feels anger towards YY.

Note that only one rule was given to Genesis, and neither of the events described in the rule's antecedent—resting or being disturbed—are explicitly mentioned in the story. However, to the analyzer of this story, it may be obvious that actions such as watching television and lying down are things one does to relax, and that waking

someone up is a form of disturbing that person. This is all commonsense knowledge, and with ConceptNet's help, Genesis can apply this knowledge to the story to comprehend it in a much more complete manner.

The elaboration graph for Genesis's ASPIRE-assisted analysis of this story is shown in Figure 7.3. ASPIRE events and connections are represented in the elaboration graph using magenta: any casual connection formed by ASPIRE is drawn with a magenta line, and any inferred event that ASPIRE added to the story is shown in a magenta box. If the formation of an ASPIRE casual connection required ConceptNet knowledge, it is dotted. ASPIRE makes connections without the help of ConceptNet when it connects a goal completion action ("Fred watches television") to the explicit goal completion ("Fred relaxes") on its own. It can also make connections using Genesis rules. In these cases, the causal connection in the elaboration graph is solid rather than dotted.

When Genesis analyzes this story without the help of ASPIRE and ConceptNet, it fails to make any causal connections, even when CSERM is turned on. This is not entirely surprising because the ruleset here is so sparse and does not relate to any explicit story events, but it highlights the power of ASPIRE and ConceptNet.

As evident from the elaboration graph, ASPIRE greatly increases Genesis's comprehension of this story. Genesis now makes connections such as Fred bought beer because he wanted to relax and lay down because he wanted to rest, and his day of working hard at the office caused these desires to relax and rest. Because ASPIRE added the explicit goal completion event "Fred rests" to the story and CSERM sees that "disturb" and "wake up" are similar, Genesis is now able to apply the sole rule it was given to deduce that Fred became angry when John woke him up with his music. ASPIRE then uses this deduction to connect Fred's anger to him sighing, taking a walk, and calming down. These later events were motivated by Fred's goals of cooling off and releasing energy. Some of the justifications ASPIRE provides for its ConceptNet-assisted connections are shown in Figure 7.4.

Let's consider an additional demonstration of ASPIRE's abilities, this time on a story which feels a little more likely to be naturally encountered. The following is a

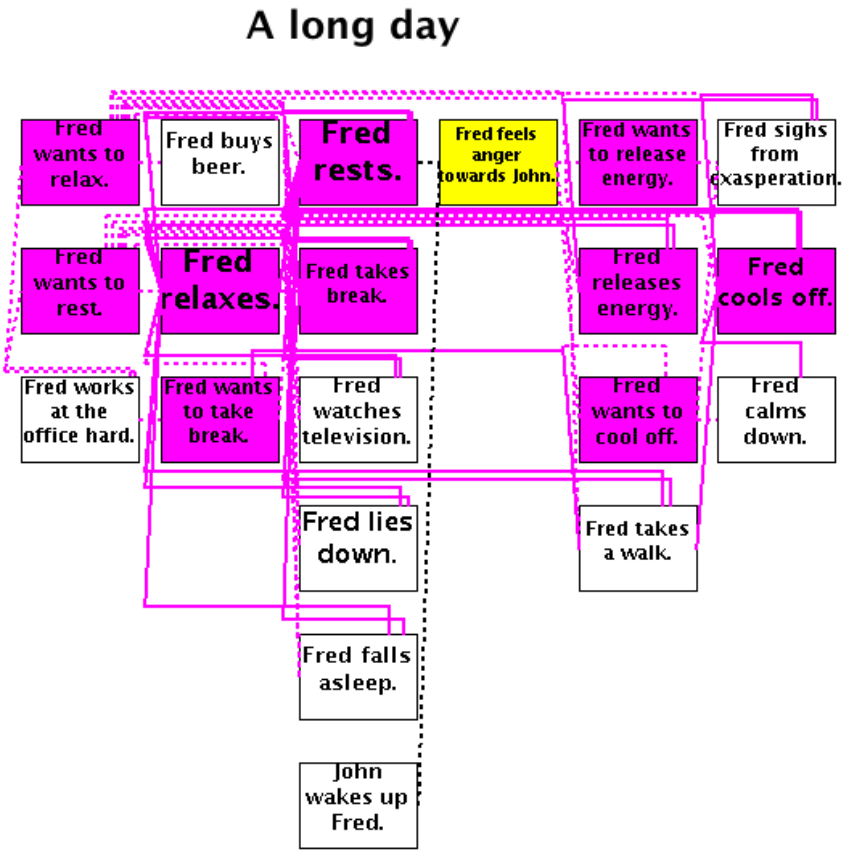

Figure 7.3: The elaboration graph generated by Genesis for "A Long Day." All ASPIRE inferred events and connections are shown in magenta. Connections that used ConceptNet knowledge are dotted. Without ASPIRE and ConceptNet, Genesis fails to make any causal connections.

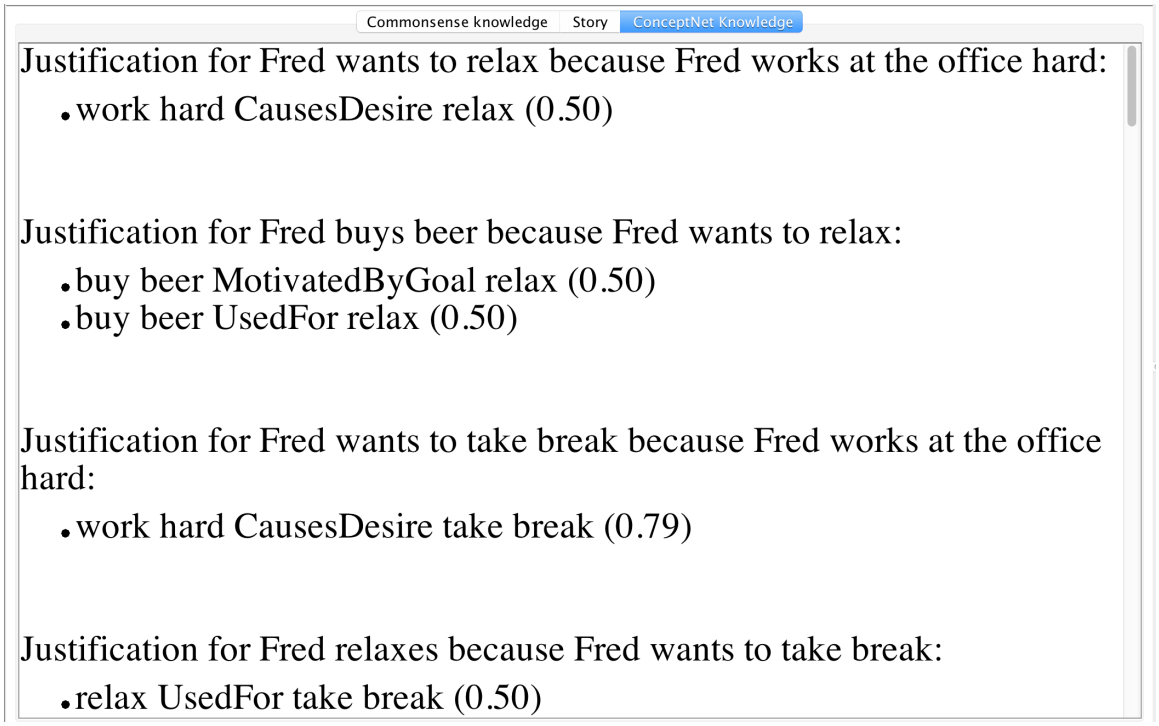

Figure 7.4: Justifications produced by ConceptNet for "A Long Day." Each justification names a causal connection and the ConceptNet assertion(s) and associated score(s) that motivated forming the connection.

simple retelling of the Mexican-American war, a conflict which occurred in the 1840s:

> The United States is a country. Mexico is a country. The year is 1846. Manifest Destiny is popular in the United States. The United States has ambition, has greed, and wants to gain land. Mexico and the United States disagree over borders. The United States move into the disputed territory. Mexican forces attack the United States. The United States declares war on Mexico. Winfield Scott leads the United States army. The United States battles Mexico. The United States triumphs by capturing Mexico City. The United States defeats Mexico and wins the war. The Treaty of Guadalupe Hidalgo officially ends the war.

This summary of the war is plain, brief, and high-level, but resembles an elementary school textbook chapter summary. This time, let us explore how Genesis analyzes a story using ASPIRE when it is not given any rules. This means CSERM,

by its very design, cannot help here because Genesis lacks any rules to compare story events against for similarity. There is one goal explicitly stated in the story—"The United States wants to gain land"—but Genesis is not told how this relates to any of the other story events. Again, without the ASPIRE system enabled, Genesis does not make any causal connections. The elaboration graph of Genesis's analysis of the story using ASPIRE is shown in Figure 7.5.

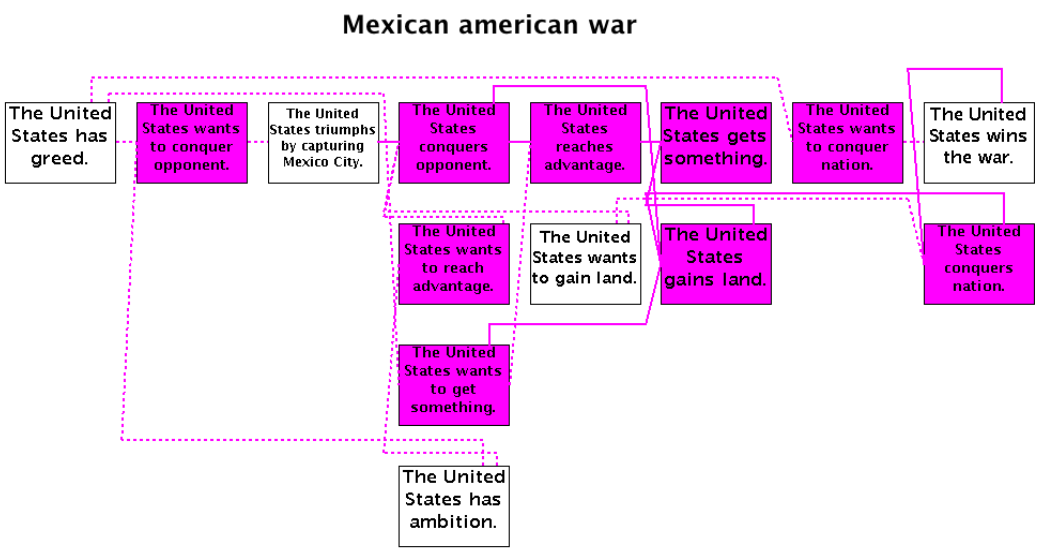

Figure 7.5: The elaboration graph produced by Genesis for the Mexican-American war story. All ASPIRE inferred events and connections are shown in magenta. Connections that used ConceptNet knowledge are dotted. Genesis was not given any user-defined rules to help analyze this story, instead relying entirely on ConceptNet.

ASPIRE allows Genesis to connect the United States' ambition and greed to its actions taken against Mexico. Genesis's inferences depend on its commonsense knowledge and are just one possible interpretation of the story's events, and the English generation of some of its inferences is not perfect. Still, Genesis's inferences are salient because the effective application of relevant knowledge, not the knowledge itself or the English generation, is the focus of my work and the example. ASPIRE helps Genesis connect the United States' ambition to its desire to "conquer [its] opponent," which it completes when it triumphs by capturing Mexico City. Conquering its opponent allows it to accomplish its goal of gaining land. Similarly, the United States "conquers [a] nation" by winning the war, and its goal of conquering a nation stemmed from its

greed. Conquering a nation also helps it gain land. ASPIRE is not currently able to recognize that the conquered opponent and conquered nation refer to the same object and resolve these two events together; section 8.1 discusses this further. The country's greed makes it want to get something, and this goal is accomplished when the United States gains land, an example of an ASPIRE inference enabling it to make an additional inference. None of the goal concepts (e.g. "conquer opponent," "get something") besides "gain land" appear in the story text, but Genesis has inferred that these concepts are relevant and has connected these inferences to explicit story events. Note that the story does not explicitly state any causal connections, so all causal connections Genesis forms are due to the application of commonsense knowledge. Genesis was able to make all of these connections and inferences without any user-specified rules; instead, it's relying just on ConceptNet knowledge to connect the United States' traits, goals, and actions.

7.3 Interaction with Other Genesis Capabilities

In this chapter I've been focusing on the elaboration graph to show how CSERM and ASPIRE increase the number of causal connections. However, causal connections are just the beginning. Causal connections are the base of the majority of Genesis's features because they form its reasoning substrate, but these features, not the connections themselves, are the more visible and exciting part of the system. The additional causal connections CSERM and ASPIRE form improve the performance of many Genesis features even though the user has given Genesis less information through specifying rules. As an example, let us examine how Genesis answers some questions about the Mexican-American war story from section 7.2. Three questions and Genesis's associated answers are displayed in Figure 7.6.

When Genesis is asked "How did the United States gain land?", it responds with two story elements that were never explicitly stated, but it inferred using its common sense. While the grammar is not perfect, the content of the answers is the focus. When Genesis is asked about some of these inferred elements, it responds with

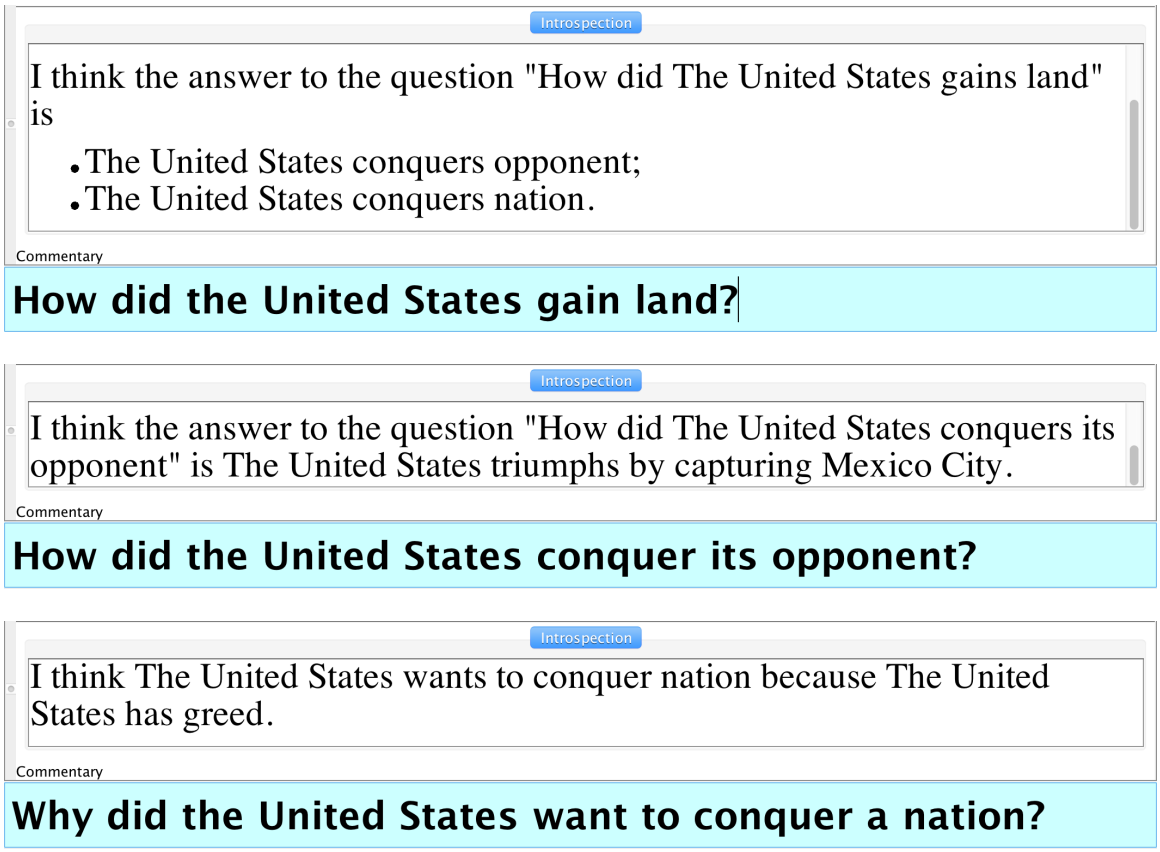

Figure 7.6 (for reference only; identical to Figure 1.1): A series of questions and Genesis answers regarding the Mexican war story. Genesis answers the first question with two story elements that were never explicitly stated, but it inferred using its common sense. The second and third questions ask Genesis about these inferences, and it backs them up with explicit story elements. Genesis's grammar in its English generation is not perfect, but the content is salient and the inferences are interesting.

further explanation, citing specific story events to back up its inferences. Without ASPIRE or any ConceptNet knowledge, Genesis would form zero causal connections and therefore couldn't answer any "Why" or "How" questions about this story! The causal connections CSERM and ASPIRE contribute are of great value to Genesis because so many of its capabilities rely on the connections formed between story events.

7.4 Quantifying Gained Commonsense Knowledge

It's difficult to precisely express the amount of ConceptNet knowledge Genesis can now access in terms of equivalent Genesis rules for several reasons. Firstly, both CSERM and ASPIRE use cutoff values to test if a piece of ConceptNet knowledge is certain enough to be acted upon. Therefore, the amount of actualized commonsense knowledge depends on the settings of these cutoff values. However, I assume for the purpose of this analysis that the cutoff value is 0.3 for ASPIRE, a value which I have experimentally found to produce satisfactory results. I report results regarding CSERM with two cutoff values, 0.33 and 0.5, because this feature is especially sensitive to the choice of cutoff.

Part of speech also complicates analysis. CSERM currently only detects when a rule verb and story event verb are similar. ASPIRE primarily extracts verbs and verb phrases from story events, although it can extract noun direct objects in some cases as discussed in section 6.3.2. Because these two systems focus on verbs when interacting with ConceptNet, not all of ConceptNet's relevant knowledge is accessed; instead, Genesis currently primarily uses only the relevant knowledge that has verbs or verb phrases as concepts. Therefore, it's necessary to perform part-of-speech analysis on the relevant ConceptNet knowledge when quantifying how much knowledge Genesis has gained.

Performing part-of-speech analysis on ConceptNet concepts is somewhat approximate. There is no straightforward, entirely reliable way to computationally determine the part of speech of a concept from its broader context in an assertion because as-

Rule Amplification Factors		
Concept	Cutoff Value of 0.33	Cutoff value of 0.5
Harm	144	38
Observe	114	17
Love	116	16
Sprint	44	6

Table 7.1: Rule amplification factors for several concepts under different cutoff values. General concepts benefit more from CSERM.

sertions are small ideas, not full sentences. Moreover, a concept may have multiple possible parts of speech, and an assertion could apply to more than one of these senses (e.g. "love Causes Desire marry" is true for both the noun and verb "love"). Therefore, I take a rather crude approach and treat any concept that can be a verb as a verb for the purposes of quantifying gained ConceptNet knowledge. These tallies are not meant to be precise counts, but rather ballpark estimates which show the magnitude of what Genesis has gained.

7.4.1 Common sense enabled rule matching

I define the "rule amplification factor" for a verb v as the number of story event verbs that CSERM would detect as similar to v in a rule, thereby amplifying the rule to apply in many more contexts. I calculated the rule amplification factor that CSERM provides for four concepts: "harm," "observe," and "love," and "sprint." This calculation describes the number of concepts that occur in both ConceptNet 4 and 5 whose similarity to the base concept surpasses the cutoff value. As described in Chapter 4, the similarity score I use is a function of ConceptNet 4's similarity and ConceptNet 5's relatedness. I limited the concepts to just verbs because CSERM currently only detects similar verbs. My approximate part-of-speech analysis perfomed surprisingly well for this calculation; inspecting the similarity results shows that true verbs make up nearly all of the results. These scores also only consider similarity between concepts that WordNet does not match together, as Genesis already uses WordNet for rule matching and I'm interested in exploring what ConceptNet adds to Genesis.

The rule amplifications are shown in Table 7.1. Using a cutoff value of 0.33

produces amplification factors of around 100, while a cutoff of 0.5 lowers the amplification factor to around 25. General concepts such as "harm" have higher rule amplification factors, while more specific ones such as "sprint" do not benefit as much from CSERM. Many of the similar concepts at each cutoff value are highly relevant. For instance, similar concepts to harm at a cutoff of 0.33 include "deceive," "threaten," "rob," "anger," "attack," "torture," "dishonor," "murder," and "embarrass." Of course, many of the results do not meet this quality level. For instance, one of the similarity matches for "love" is "hate." The two are similar in many ways just as "hot" and "cold" are similar, but it's unlikely that a rule describing love is applicable to a story event describing hate. Other similarity results for "love" include "feel," "forgive," and "cuddle," all concepts that, while somewhat relevant, are not guaranteed to produce a satisfactory common sense enabled rule match. The cutoff value controls the quality of matches—higher cutoff values produce higher percentages of quality matches, but the number of total matches falls.

7.4.2 ASPIRE

ASPIRE uses ConceptNetClient to query ConceptNet for information about goals, specifically using the "Causes Desire" relation to infer goal causation and the "Motivated By Goal," "Used For," and "Has Subevent" relations to infer an action's contribution to a goal. Of the approximately 225,000 assertions ConceptNet 4 contains, around 20,000 of these are relevant to the goal information ASPIRE requests. More precisely, with "goal action" defined as an action taken that does indeed contribute to a goal, there are about 20,000 assertions that either:

- describe a cause of a goal for which ConceptNet knows at least one goal action, or
- describe goal actions of goals for which ConceptNet knows at least one cause.

I require knowing at least one cause and at least one action for each goal in this tally because ASPIRE must see both these sides of the goal in order to infer the goal

and influence story processing; if ConceptNet only knows one of these sides, ASPIRE could never use this information to make an inference in a story.

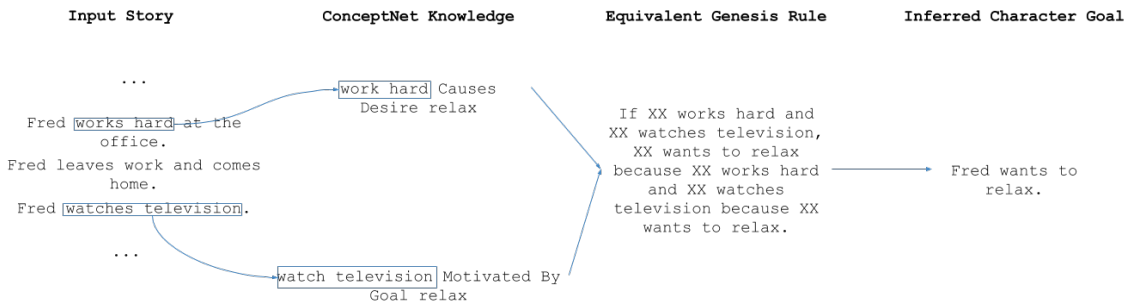

Figure 7.7: An example of an ASPIRE equivalent Genesis rule. Although equivalent Genesis rule are never explicitly constructed, formulating ASPIRE's use of ConceptNet knowledge as an equivalent Genesis rule is useful for comparative purposes. ASPIRE's use of ConceptNet currently incorporates about 30,000 equivalent Genesis rules into story processing.

However, these 20,000 assertions are equivalent to a greater number of equivalent Genesis rules. Equivalent Genesis rules are never explicitly constructed by ASPIRE, but it's helpful to frame ASPIRE's use of ConceptNet's knowledge in terms of equivalent Genesis rules for comparative purposes. ASPIRE requires seeing a cause of a goal and a goal action before inferring the presence of the goal, as shown in Figure 7.7. Therefore, there's a multiplicative effect between causes of goals and goal actions when calculating the number of equivalent Genesis rules because every cause of a goal is paired with every goal action. With this in mind, ConceptNet's knowledge contains approximately 90,000 equivalent goal-related Genesis rules under ASPIRE's framework for goal inference.

Although ConceptNet provides 90,000 rules relevant to ASPIRE, it cannot currently make effective use of all of them. ASPIRE primarily extracts verbs and verb phrases from Genesis entities, not nouns or noun phrases, so it's unlikely to use an assertion such as "humor Causes Desire laugh" because it requires extracting the noun "humor" as a concept. However, ASPIRE could be extended to more completely extract concepts and take advantage of all the assertions ConceptNet has to offer. Therefore, ASPIRE's use of ConceptNet effectively adds 90,000 potential commonsense rules to Genesis, but only 30,000 are verb-based and currently incorporated

into story processing. ConceptNet contains approximately 1,000 assertions describing causes of goals where both concepts are verbs or verb phrases, and ConceptNet contains approximately 11,0000 assertions describing actions taken towards completing a goal where both concepts are verbs or verb phrases.

Note that this result is only a ballpark approximation due to systematic overestimation and underestimation in different parts of the computation. I did not tally the restricted noun-related knowledge that ASPIRE is able to take advantage of as discussed in section 6.3.2, but I also rather generously treated all concepts that can be verbs as verbs in every assertion of interest.

Chapter 8

Limitations and Future Work

While ConceptNet, common sense enabled rule matching (CSERM), and ASPIRE certainly contribute a lot to Genesis, there are some important limitations to these additions. Several of these limitations stem from ConceptNet's imprecise semantics and limited depth of knowledge. ConceptNet is a great resource for giving computing applications the common sense necessary for interacting with humans, and its union with Genesis is a significant step toward artificial general intelligence. However, ConceptNet is not a panacea. Common sense is wide, ambiguous, and imprecise, making it hard to computationally pin down. Although ConceptNet provides important context that enables its applications to reach deeper understandings, the depth of the knowledge ConceptNet captures is still limited.

More specifically, there is a need for disambiguating concepts retrieved from ConceptNet, ASPIRE requires a more robust method of determining if an action completes a goal or if it is just a step towards goal completion, and CSERM can introduce a significant slowdown. There are also numerous ways of extending this work. I highlight ways that Genesis can interpret ConceptNet scores more methodically, can use more ConceptNet knowledge, and can expand its commonsense knowledge beyond ConceptNet. Note that other difficulties like parsing complex sentences must be addressed before Genesis can be applied to a large corpus of naturally occurring text, but I do not discuss those here because they are not directly relevant to my work.

8.1 Limitations

Unfortunately, ConceptNet does not contain any information about the sense in which a concept is being used. For instance, "dog Has Property bark" and "tree Has Property bark" are both assertions in ConceptNet 4, each referring to a different meaning of "bark." This means Genesis could apply ConceptNet knowledge in a situation where it's not appropriate because it's using a word in a different sense than the knowledge describes. This issue is subtly present in the Mexican-American war story introduced in section 7.2. The United States reaches an advantage because it conquers its opponent, and the country's greed gives it a goal of getting something. ASPIRE sees the United States reaching an advantage as completing its goal of getting something because ConceptNet provides the knowledge "reach Used For get." However, this justification is not valid because "reach" is being used in its "succeed in achieving" definition in "reach advantage," while the justification applies to its "stretch out an arm in a specified direction in order to touch or grasp something" definition (Reach., 2017). This misapplication is pretty minor, but the issue can manifest itself in much more unfortunate ways. WordNet may be able to help with disambiguating the commonsense knowledge because it provides glosses that describe the sense of a word. Combining these glosses with ConceptNet concepts is not trivial, so this may remain as a fundamental limitation of ConceptNet knowledge and serve as a motivation to explore other commonsense knowledge bases. Disambiguation with WordNet has proven successful in the past with previous versions of ConceptNet, though, so the approach does show promise (Chen & Liu, 2011).

ASPIRE is also unable to detect when two concepts received from ConceptNet refer to the same object in a story and resolve these concepts together. Genesis's analysis of the Mexican-American war story introduced in section 7.2 has an example of this as well. ASPIRE connects the United States' greed to its desire to conquer a nation as well as the United States' ambition to its desire to conquer its opponent. Both of these goals are completed by actions the United States takes in the story, leading to the instantiation of both "The United States conquers [its] opponent" and

"The United States conquers [a] nation." Of course, the conquered opponent and conquered nation both refer to Mexico, so while the knowledge about both conquering opponents and conquering nations is relevant in this case, ASPIRE would ideally recognize that these two goal instantiations are somewhat redundant. Unpacking the received concepts and keeping track of mappings between story entities and concepts could help address this problem, but this issue is more cosmetic and less pressing than the other limitations because ASPIRE is still applying its commonsense knowledge correctly. Querying ConceptNet for the similarity of "conquer nation" and "conquer opponent" could also be helpful in solving this problem.

The ASPIRE system operates using an important simplifying assumption which does not always hold: if a goal has one or more actions taken towards completing the goal, the goal has been completed. As an example, suppose ASPIRE has formed the candidate character goal "Matt wants to kiss Jackie." ASPIRE would connect the event "Matt embraces Jackie" to this goal because ConceptNet provides the knowledge "kiss Has Subevent embrace." Because the goal has had an action taken towards it, ASPIRE would instantiate both the inferred goal "Matt wants to kiss Jackie" and the inferred goal completion "Matt kisses Jackie." Instantiating the goal completion at this point is premature, though, and is incorrect if the story later indicates that Matt did not kiss Jackie after all.

The problem worsens when ASPIRE makes additional inferences based on an incorrect inferred goal completion, allowing ASPIRE to form long chains of faulty inferences stemming from one or many misapplications of commonsense knowledge. For instance, suppose Genesis is given the following version of the "A Long Day" story from section 7.2, this time with an alternate ending:

> Fred is a person. John is a person. John is Fred's roommate. Fred works hard at the office. Fred leaves work. Fred buys beer. Fred comes home. Fred drinks beer and watches television. Fred lies down. John comes home. John plays loud music and wakes up Fred. Fred fights with John over his music. Tempers flare. John apologizes and they drink beer together.

And the same sole rule as earlier:

- If XX is resting and YY disturbs XX, then XX feels anger towards YY.

ASPIRE arrives at a very different interpretation of the story than before, as shown in Figure 8.1. Fred's anger and his fighting lead ASPIRE to incorrectly infer and complete several additional realized goals for Fred, including killing a person, joining the army, and dying; the complete set of ConceptNet justifications for this story can be found in the Appendix. These chains of inferences are all due to ASPIRE's coarse method of detecting goal completion. ASPIRE believes Fred kills a person because ConceptNet tells it that anger causes the goal of killing a person and fighting is evidence of trying to kill a person. This knowledge alone is not sufficient for concluding Fred does indeed kill a person, though—fighting does not necessitate any killing.

It would be much better if ASPIRE knew, for any given goal, which sorts of actions complete that goal vs. which sorts of actions merely contribute to it. This requires further nuance in ASPIRE's knowledge, which it obtains through Genesis rules and ConceptNet. ConceptNet 4 does not provide this sort of nuance, so exploration into alternative commonsense knowledge bases is necessary to address this shortcoming. In the meantime, a possible defensive fix is to develop a setting for ASPIRE that allows the system to infer goals, but not assume and instantiate their completion.

Alternatively, the previous two issues of concept disambiguation and goal completion could be more simply solved by incorporating human interaction. Genesis could present a ConceptNet-assisted interpretation of a story to the user, and the user could inform it if it made any mistakes and why. In this way, Genesis could augment its commonsense knowledge, collecting the information it needs to be even more accurate directly from humans. After all, humans are taught much of what they know directly from other humans, especially at an early age. Humans could instruct Genesis in a similar manner, giving it the information it needs to correct its mistakes.

One final limitation is the slowdown that CSERM can introduce. If there are a large number of rules, Genesis's processing can slow significantly because of the

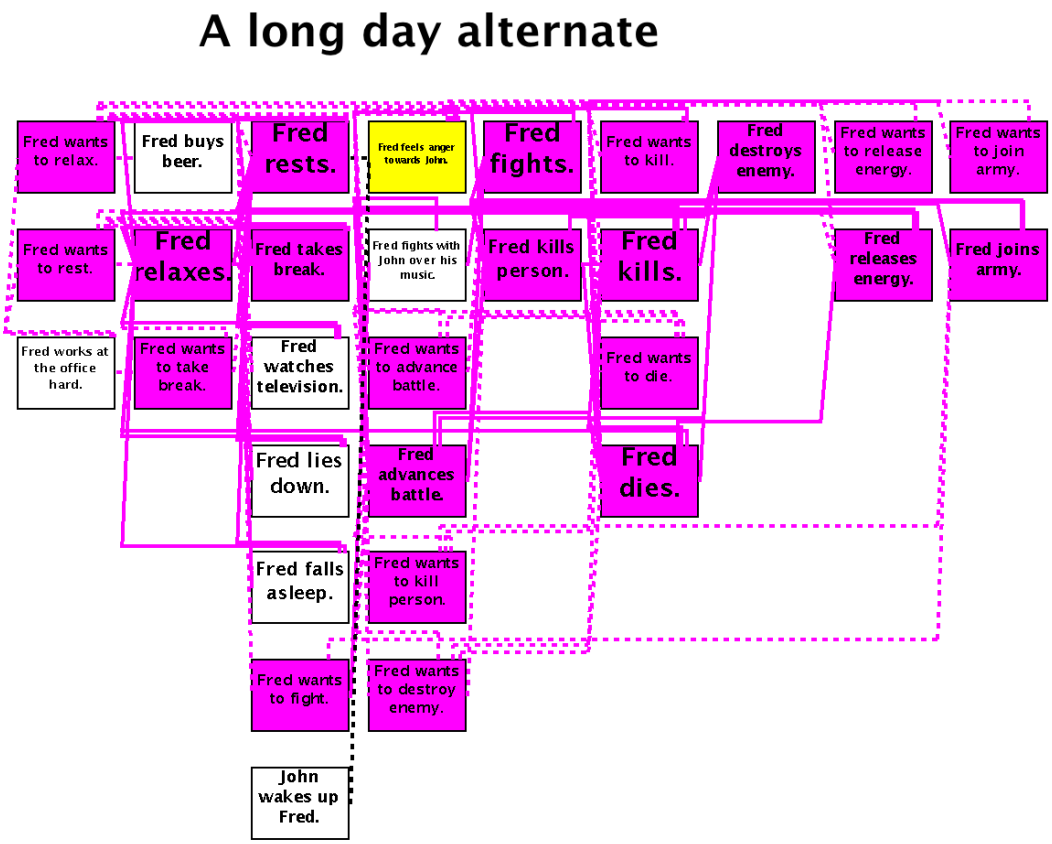

Figure 8.1: The elaboration graph produced by Genesis for the alternate version of "A Long Day." Here, the understanding ASPIRE reaches is deeply flawed, caused by a series of misapplications of ConceptNet knowledge.

added overhead that similarity checks add. ConceptNet 5's rate limiting worsens the problem. ConceptNet 5 only allows sixty requests per minute, and if Genesis exceeds this limit, it must pause and wait for the limit to expire before it can send its next query. Because similarity results are cached, this issue only occurs the first time a story is processed. Still, the slowdown is a significant annoynace when applying CSERM to longer stories with large rulesets. ConceptNet 5's relatedness data is available for download, so this issue could be solved by serving relatedness requests using local data rather than by querying the server.

8.2 Future Work

All knowledge provided by ConceptNet is scored, with higher-confidence assertions receiving higher scores. These scores are useful because they allow the user of the knowledge to compare assertions and control more finely when an assertion is "true enough" to be actionable. Currently, CSERM and ASPIRE use score cutoffs to filter assertions, acting upon only those with scores greater than or equal to the cutoff. CSERM is more sensitive to its cutoff value than ASPIRE because similarity scores are continuous and the scores of the assertions ASPIRE receives have far less variability. These cutoffs have been experimentally chosen at the moment, but choosing them dynamically would increase the robustness of the implementation. More specifically, the cutoff could adjust as the story is being read in response to the story's perceived coherence. I have completed work prior to this thesis that allows Genesis to assess how coherent it finds a story by using metrics such as the number of connected components in a graph of the causal relationships between story entities. The cutoff could decrease if Genesis finds the story relatively incoherent so that it could make more casual connections; conversely, the cutoff could increase if Genesis thinks it's understanding the story pretty well, as additional commonsense causal connections may just result in noise. Such a design does assume a universal expected level of coherence for every story Genesis reads, as Genesis makes changes to its internal processing to try to reach this coherence level. Although this assumption may not hold for all stories, it's likely true for a large majority, so I believe this approach to be worthy of exploration as a valuable way to avoid hard-coded cutoffs.

Finally, there are a number of ways to extend Genesis's commonsense knowledge. A first target is using more of ConceptNet's knowledge. CSERM only uses ConceptNet's similarity information, and ASPIRE only uses the "Causes Desire," "Motivated By Goal," "Used For," and "Has Subevent" relations. However, there are numerous other relations that would provide valuable information to Genesis, with "Causes" being the most promising one. ConceptNet assertions with "Causes" as the relation describe general causal relationships (e.g ConceptNet 4 has assertions such as "tickle

Causes laughter" and "read Causes learn"). This knowledge could directly wire into Genesis's story processing so Genesis could identify general causal relationships in the story on its own, moving beyond the goal-oriented causal connections that ASPIRE forms.

It should be noted that using ConceptNet's knowledge in a standalone manner with no dependence on user-defined rules requires careful thought. ASPIRE uses ConceptNet in this powerful fashion, and section 6.3.2 discusses the rather involved method I developed for extracting concepts from goal-related Genesis entities and incorporating goal-related ConceptNet knowledge. I arrived at this method through studying the structure of ConceptNet goal-related knowledge, observing common patterns, and constructing procedures that take advantage of these patterns as safely as possible. A similar process is necessary to incorporate other domains of ConceptNet knowledge into Genesis, especially because ConceptNet knowledge is in a fundamentally different format than a Genesis rule (its lack of bindings is discussed in section 6.3.2). The connection I've established between the two systems easily supports integrating new types of ConceptNet information, but determining exactly how new types of information should be integrated requires programmer effort.

ConceptNet is a valuable commonsense knowledge base which has proved very effective, but it's only the beginning. I mention several other commonsense knowledge bases in Chapter 9 which are worth exploring such as Cyc and WebChild. ConceptNet does have its limitations as discussed in section 8.1, and incorporating other knowledge bases could address some of these shortcomings while simultaneously adding more knowledge for Genesis to reference as it processes stories. ConceptNet is a great step towards capturing the common sense necessary for building artificial general intelligence because it gives applications real-world context and knowledge. Connecting Genesis to ConceptNet is truly transformative because it expands the amount of information accessible to Genesis useful for processing stories tremendously. Still, although ConceptNet provides important context that enables its applications to reach deeper understandings, ConceptNet's limited depth of knowledge is a reason to explore alternatives in an effort to apply common sense in a more robust manner.

Chapter 9

Related Work

9.1 Machine Learning Approaches

The field of machine reading, or the application of machine learning techniques to natural language understanding, has grown in popularity recently. Machine learning natural language efforts have traditionally focused on lower-level problems such as sentiment analysis or topic detection rather than larger problems such as extracting causality from text, the focus of my work. However, the emergence of machine reading signifies a renewed interest in higher levels of comprehension.

Machine reading is advancing, supported by new benchmark datasets such as the Stanford Question Answering Dataset (SQuAD) (Rajpurkar, Zhang, Lopyrev, & Liang, 2016). Facebook has released a set of twenty machine reading question answering tasks, and Google DeepMind has published a large question answering dataset designed to test reading comprehension (Hermann et al., 2015; Weston et al., 2015). All three of these datasets consist of passages, associated questions, and correct answers, and these benchmarks have spurred a flurry of activity in neural network development and tuning (Hill, Bordes, Chopra, & Weston, 2015; Kumar et al., 2015; *SQuAD leaderbord*, n.d.). However, the answer for every question in all these datasets is always explicitly stated in the input text (besides when the question has a yes or no answer, in which case the story explicitly contains all relevant information needed to answer the question). These datasets require little inference and

commonsense knowledge, as all questions can be answered by combining information in different sections of the input text. There is no flexible knowledge representation, no explicit commonsense reasoning, and, I argue, a rather shallow level of understanding because the network never needs to respond to a question with an answer that is not explicitly stated in the story. These datasets are a good first step towards more general intelligence because they refocus natural language processing (NLP) research on higher-level reasoning tasks, but humans often leave commonsense knowledge implicit when they communicate. Therefore, any system designed to fully understand human communication must incorporate large amounts of background commonsense knowledge, and these datasets do not address this aspect of natural language understanding.

Another relevant area of NLP research is recognizing textual entailment (RTE). RTE tasks a model with deciding if two sentences have an entailment (one can be inferred from the other), contradictory, or independent relationship. This task is important because inference is of great importance to story understanding. Bowman, Angeli, Potts, and Manning (2015) have published a popular dataset of sentence pairs labeled with their entailment status, and this dataset has generated considerable interest in tackling the RTE problem (Rocktäschel, Grefenstette, Hermann, Kočiskỳ, & Blunsom, 2015). RTE is valuable because of its focus on inference, but little work has gone into making inferences from a sentence or story rather than evaluating an inference from already-curated candidate pairs. When attempting to understand a story, a reader is not given potential inferences paired with corresponding story text for evaluation; they must generate their own inferences from the story's events, as my work does. Kolesnyk, Rocktäschel, and Riedel (2016) are nearly alone in their work on generating inferences from a given source sentence rather than evaluating the entailment of previously defined pairs. They use Bowman et. al's dataset to train a neural network to generate an inference from a starting sentence. However, the resulting inferences are not very exciting. Examples include inferring "A man is driving a car" from the starting sentence "He is driving a beautiful red car" and "There are people looking at the sky" from the starting sentence "Space travellers

gaze up and strain to find the blue dot in their sky." While these inferences are correct and could at times be potentially useful to a story-understanding system such as Genesis, they are rather plain. My work enables Genesis to make more elaborate goal-related inferences through ASPIRE's use of ConceptNet, and it's not clear that Kolesnyk et. al's approach could produce similarly nuanced inferences. Furthermore, Genesis is able to produce user-friendly justifications explaining the inferences it made, something machine learning approaches lack.

While limited, there have been some machine reading approaches that use background knowledge to aid the analysis of input text. Nakashole and Mitchell use a large amount of background knowledge to assist with parsing. They focus on addressing two common parsing problems in their 2016 paper, prepositional phrase attachment ambiguity and relationships between compound nouns. When faced with parsing ambiguity, they consult background knowledge bases and corpora to detect how the ambiguity should be resolved. While this approach does incorporate background knowledge, the knowledge is used for parsing purposes only, and it's hard to characterize the knowledge used as common sense. My work focuses on using background commonsense knowledge not for parsing purposes, but to enhance story understanding by inferring nontrivial elements of the story never explicitly stated, a capability machine reading has not yet developed.

Weston, Chopra, and Bordes' (2014) work with memory networks explores how a learning model can incorporate a long-term memory component. They apply such a model to a large-scale question answering task where the system was given fourteen million (subject, relation, object) triples and asked questions about them. The model was also applied to much smaller stories of approximately a dozen sentences and asked questions that required the model to combine knowledge from events early and late in the story. However, the paper does not discuss giving the model both the large assertion knowledge base and the input story together. Then, the model could be asked questions that require incorporating assertions from the knowledge base into the processing of the smaller story, similar to how my work combines commonsense knowledge with story analysis. Memory networks may not be well designed for such

a task.

Word embedding is a currently popular NLP technique for detecting the relationship between words. The idea is that every word can be represented by a high-dimensional vector, usually computed by using machine learning on a large dataset, and these vectors are called "word embeddings." The relatedness of two words can be computed by taking the dot product of these vectors. Google's Word2vec model is one popular implementation of the idea, and the ConceptNet team has also developed their own word embedding library (Mikolov, Chen, Corrado, & Dean, 2014; Speer & Chin, 2016). Word embeddings do not provide high-fidelity commonsense knowledge because they cannot explain the relationship between two concepts. ConceptNet knows assertions such as "wing Used For flight," not just that "wing" and "flight" are related. ConceptNet 5 uses word embeddings to generate relatedness scores, which common sense enabled rule matching (CSERM) incorporates. However, word embeddings were not sufficient on their own in this project, as ASPIRE requires high-fidelity commonsense knowledge beyond the numerical relationship of two concepts, and CSERM incorporates other similarity metrics besides word embeddings.

9.2 Story Understanding and Common Sense

Story understanding has been a topic of interest in artificial intelligence for decades, although popularity has recently waned. Schank and Abelson proposed in 1995 that stories are a fundamental part of human memory, knowledge, and social communication. More recently, Winston (2011) articulated a similar view in his Strong Story Hypothesis, which states that story telling and understanding are central to human intelligence.

Despite the importance of stories, there have been few attempts at applying commonsense knowledge to story understanding. Gordon (2016) recently framed the problem of forming commonsense interpretations as a process of logical abduction. He developed a model which reads a small story and chooses the more likely of two possible relevant story inferences. The model arrives at its choice by consulting 136

hand-coded probabilistic commonsense axioms to generate scored hypotheses. He applied this approach to the Triangle-COPA benchmark question set. The model performed well on the benchmark, but it's not clear how scalable the approach is. The model requires carefully constructed commonsense axioms to arrive at an answer, and Gordon does not propose a way to generate these axioms using existing knowledge bases. Commonsense knowledge can be ambiguous, contradictory, and context-dependent, so its very nature may disagree with the probabilistic approach Gordon takes. My work uses ConceptNet to gain a wide breadth of commonsense knowledge and embraces the context dependency of common sense by applying ConceptNet knowledge only when the story shows evidence that such knowledge is relevant. Furthermore, Genesis can make its own commonsense inferences without being supplied options from which to choose, whereas Gordon's model only chooses between two possible inferences.

Mueller (2003) has decomposed the problem of story understanding into building multiple models of a story, giving a program multiple representations of commonsense knowledge, and combining the two using a satisfiability solver to produce a model of the story as output. This approach has some efficiency limitations and it's not clear how well it would scale to larger stories, more common sense, or a larger variety of stories, and the codified commonsense knowledge is currently rather small in size. Mueller has since further developed his event-calculus approach (2014), but his logic-based formulation of common sense may be too rigid and constrained to be universally applicable, as common sense can be ambiguous and contradictory by its very nature.

There are several versions of ConceptNet, but I primarily used ConceptNet 4 in my work. All of ConceptNet 4's knowledge is directly collected from humans, and this knowledge can be viewed in terms of Schank, Abelson, and Winston's beliefs regarding stories. Schank and Abelson argue that much of human knowledge is story-based, and Winston argues that stories are a fundamental part of human intelligence. In this light, humans may represent their commonsense knowledge using small story snippets. I might believe the assertion "person Desires privacy" because I have mentally retained many instances of specific people, including myself, desiring privacy.

With this interpretation, ConceptNet is a database of common sense because it's a database of story snippets. This means that when Genesis uses ConceptNet's knowledge to understand a story, it's consulting story snippets collected from humans to understand similar events in the current story.

These ideas are reminiscent of Minsky's K-Lines theory of memory (1980). Minsky hypothesized that when a certain idea is remembered, the brain reactivates some of the mental states present when that idea was originally conceived. Schank and Abelson similarly contend that new events are interpreted in terms of old stories. Genesis is taking a comparable approach with commonsense knowledge when it reasons about one story by consulting common sense collected from similar stories. In Genesis's case, these stories are not from its own past experience, but from the past experiences of many different humans.

9.3 Commonsense Knowledge Bases

There are a number of other commonsense knowledge bases besides ConceptNet, such as Cyc and WebChild (Guha & Lenat, 1994; Tandon, de Melo, Suchanek, & Weikum, 2014). There have also been efforts to extend ConceptNet with content from the Web, which increases the amount of knowledge in the knowledge base by several orders of magnitude (Tandon, De Melo, & Weikum, 2011). I considered these alternatives, but chose to primarily give Genesis ConceptNet 4's knowledge for several reasons. Most readily, ConceptNet's data is freely available in a web API, while resources such as Cyc are more restricted and require a license to use. Moreover, ConceptNet's biggest strength is that its data is collected directly from humans, which allows it to capture the nuances of common sense that would be hard to codify without going directly to humans. For instance, ConceptNet has assertions such as "competitive nature Causes Desire play sports" and "wake up Has First Subevent open eye," pieces of knowledge which are rarely explicitly expressed because they are so apparent to humans. Any sort of web-crawling, NLP-based method of collecting common sense may be able to collect more knowledge, but it's not obvious that the data would be of equal quality

(Havasi et al., 2009).

Additionally, ConceptNet can numerically compute a score describing how similar two concepts are. While several other commonsense knowledge bases have some method of computing an association between concepts, it's rare for this association to capture similarity rather than relatedness. Similarity requires inspecting how the concepts are used, whereas relatedness captures how connected the concepts are. For instance, "desk" and "paperclip" are related because they're both found in an office, but they are not similar because they have very different uses. "Desk" and "table" are similar, though, because they're both used to support things. ConceptNet provides ways to query both relatedness and similarity, an attractive quality because this similarity capability is relatively unique across knowledge bases (Havasi et al., 2009).

Chapter 10

Conclusion and Contributions

Rather than focusing on one narrow problem or a small set of problems, the Genesis system tackles the more general goal of story understanding. It uses a flexible knowledge representation to represent this understanding, and Genesis is able to demonstrate and use its understanding in a wide variety of ways including question answering, summarization, hypothetical reasoning, and story alignment.

Prior to my work, Genesis had little common sense. It needed to be told all of the knowledge it should use to understand a story in the form of rules, and this requirement was a large barrier to using Genesis in a real-world setting. Many of these rules expressed commonsense knowledge, such as "If person X harms person Y, Y becomes angry," Commonsense knowledge is plain and obvious to humans but needs to be explicitly given to computers to enable them to apply the sort of common sense that humans use implicitly.

My work gives Genesis orders of magnitude more of the common sense that humans hold in their head and use so naturally that it's hard to observe. By consulting ConceptNet, a large commonsense knowledge base composed of human-submitted assertions, Genesis can make many more causal connections when reading a story with less user effort. These causal connections are incredibly valuable to Genesis because they allow Genesis to see a story as more than just a sequence of unconnected events, and many of Genesis's story analysis capabilities depend on its ability to detect the causal connections within a story.

Giving Genesis much more common sense is a notable step toward creating the sort of AI depicted in the movies: an artificial general intelligence capable of understanding human communication. Genesis's connection with ConceptNet makes applying Genesis to a large corpus of naturally occurring text more feasible. Users can now spend less time spelling out all the necessary common sense Genesis needs to understand the story, instead focusing on domain-specific rules describing niche information not universal enough to be considered common sense. This makes Genesis a better fit for applications and domains such as legal document review and understanding medical histories. Its rich set of analytical tools can now be applied with far less effort.

I have implemented two applications of ConceptNet's knowledge within Genesis, common sense enabled rule matching (CSERM) and ASPIRE, but I believe this is just the beginning. I have outlined how these two applications can be extended, and Genesis can also use other varieties of ConceptNet common sense in order to take full advantage of all the information ConceptNet has to offer. ConceptNet 4 contains 225,000 commonsense assertions. CSERM uses this collective knowledge to more flexibly apply existing user-defined rules, achieving amplification factors of ten to one hundred. ASPIRE uses thousands of goal-related assertions to infer character goals with no dependence on user-defined rules, and the ConceptNet knowledge it operates with is equivalent to approximately 30,000 rules. However, the vast majority of ConceptNet knowledge does not relate to goals, and Genesis is currently not using this knowledge to make inferences independent of user-defined rules. Therefore, much of ConceptNet's knowledge still remains untapped. Furthermore, Genesis can consult other commonsense knowledge bases with complementary strengths so that it can maximize the effectiveness of its common sense.

There are five main contributions to my work. I believe my most important contribution is demonstrating the power of combining a story-understanding system and a commonsense knowledge base. This union enables inferring relevant, nontrivial events and connections never explicitly stated in the input at a large scale, a novel capability in the space of story understanding and machine reading. However, there are several

other more specific contributions significant for the Genesis system's development. I have established an extensible and easy-to-use connection between Genesis and ConceptNet which allows Genesis to justify its inferences to the user by citing the relevant commonsense knowledge. I have implemented common sense enabled rule matching, an application of commonsense knowledge which allows Genesis to amplify its ruleset by applying rules when story events are similar to its antecedents. I have articulated a framework for inferring characters' goals from the actions they take throughout a story. Finally, I have implemented ASPIRE and incorporated ConceptNet knowledge so that Genesis can reach a complete understanding of goal-oriented stories with a significantly reduced dependence on user-defined rules.

References

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Chen, J., & Liu, J. (2011). Combining ConceptNet and WordNet for Word Sense Disambiguation. In *IJCNLP* (pp. 686–694).

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., ... Welty, C. (2010). Building Watson: An overview of the DeepQA project. *AI Magazine*, *31*(3), 59–79.

Gentner, D., & Markman, A. B. (1997). Structure mapping in analogy and similarity. *American Psychologist*, *52*(1), 45.

Gordon, A. S. (2016). Commonsense Interpretation of Triangle Behavior. In *AAAI* (pp. 3719–3725).

Guha, R. V., & Lenat, D. B. (1994). Enabling agents to work together. *Communications of the ACM*, *37*(7), 126–142.

Havasi, C., Speer, R., Pustejovsky, J., & Lieberman, H. (2009). Digital intuition: Applying common sense using dimensionality reduction. *IEEE Intelligent systems*, *24*(4).

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching Machines to Read and Comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28* (pp. 1693–1701). Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf`

Hill, F., Bordes, A., Chopra, S., & Weston, J. (2015). The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *arXiv preprint arXiv:1511.02301*.

Holmes, D., & Winston, P. (2016). Story-enabled hypothetical reasoning. In *Advances in Cognitive Systems.*

Jackson Project. (n.d.). Retrieved from `https://github.com/FasterXML/jackson` ([Online; accessed 2017-05-19])

Katz, B. (1997). Annotating the World Wide Web using Natural Language. In *Computer-Assisted Information Searching on Internet* (pp. 136–155).

Kenrick, D. T., Neuberg, S. L., Griskevicius, V., Becker, D. V., & Schaller, M. (2010). Goal-driven cognition and functional behavior the fundamental-motives framework. *Current Directions in Psychological Science*, *19*(1), 63–67.

Kolesnyk, V., Rocktäschel, T., & Riedel, S. (2016). Generating Natural Language Inference Chains. *arXiv preprint arXiv:1606.01404*.

Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., ... Socher, R. (2015). Ask me anything: Dynamic memory networks for natural language processing. *CoRR, abs/1506.07285*.

Langley, P. (2011). The changing science of machine learning. *Machine Learning*, *82*(3), 275–279.

Levy, S. (2017, May). *The iBrain is Here: An exclusive inside look at how artificial intelligence and machine learning work at Apple.* Retrieved from `https://backchannel.com/an-exclusive-look-at-how-ai-and-machine-learning-work-at-apple-8dbfb131932b` ([Online; posted 24-August-2016])

Liu, H., & Singh, P. (2004). ConceptNet: A Practical Commonsense Reasoning Toolkit. *BT Technology Journal*, *22*, 211–226.

Locke, E. A., & Latham, G. P. (2006). New directions in goal-setting theory. *Current Directions in Psychological Science*, *15*(5), 265–268.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2014). *word2vec.* accessed 2014-04–15. https://code.google.com/p/word2vec.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39–41.

Minsky, M. (1980). K-Lines: A Theory of Memory. *Cognitive Science*, *4*(2), 117–133.

Minsky, M. (1988). *Society of Mind.* Simon and Schuster.

MTV Over the Line. (n.d.). Unpublished raw data.

Mueller, E. T. (2003). Story understanding through multi-representation model construction. In *Proceedings of the HLT-NAACL 2003 workshop on Text Meaning-Volume 9* (pp. 46–53).

Mueller, E. T. (2014). *Commonsense reasoning: an event calculus based approach.*

Morgan Kaufmann.

Noss, J. M. (2017). Who Knows What? Perspective-Enabled Story Understanding.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1717–1724).

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Reach. (2017). *Oxford English Dictionary*. Oxford University Press.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kočiskỳ, T., & Blunsom, P. (2015). Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Schank, R. C., & Wilensky, R. (1977). A goal-directed production system for story understanding. *ACM SIGART Bulletin*(63), 72–72.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Hassabis, D. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, *529*(7587), 484–489.

Singh, P. (1996). Why AI Failed – The Past 10 Years.

Singh, P. (2002). The Public Acquisition of Commonsense Knowledge.

Singh, P., Barry, B., & Liu, H. (2004). Teaching machines about everyday life. *BT Technology Journal*, *22*(4), 227–240.

Speer, R. (2007). *Learning common sense knowledge from user interaction and principal component analysis* (Unpublished master's thesis). Massachusetts Institute of Technology.

Speer, R., & Chin, J. (2016). An ensemble method to produce high-quality word embeddings. *arXiv preprint arXiv:1604.01692*.

Speer, R., & Havasi, C. (2012). Representing General Relational Knowledge in ConceptNet 5. In *LREC* (pp. 3679–3686).

SQuAD leaderbord. (n.d.). `https://rajpurkar.github.io/SQuAD-explorer/`. (Accessed: 2017-05-19)

Stork, D. G. (1997). *HAL's Legacy: 2001's Computer as Dream and Reality*. MIT Press.

Tandon, N., de Melo, G., Suchanek, F., & Weikum, G. (2014). WebChild: Harvesting

and Organizing Commonsense Knowledge from the Web. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining* (pp. 523–532).

Tandon, N., De Melo, G., & Weikum, G. (2011). Deriving a Web-Scale Common Sense Fact Database. In *AAAI.*

Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., & Mikolov, T. (2015). Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Weston, J., Chopra, S., & Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

Winston, P. H. (2011). The Strong Story Hypothesis and the Directed Perception Hypothesis.

Winston, P. H. (2014). *The Genesis Story Understanding and Story Telling System: a 21st Century Step Toward Artificial Intelligence* (Tech. Rep.). Center for Brains, Minds and Machines (CBMM).

This work includes data from ConceptNet 4 and ConceptNet5. ConceptNet 5 was compiled by the Commonsense Computing Initiative. ConceptNet 5 is freely available under the Creative Commons Attribution-ShareAlike license (CC BY SA 4.0) from http://conceptnet.io. The included data was created by contributors to Commonsense Computing projects, contributors to Wikimedia projects, Games with a Purpose, Princeton University's WordNet, DBPedia, OpenCyc, and Umbel. ConceptNet 4 is available under the GNU General Public License at https://github.com/commonsense/divisi2/. When this work references specific ConceptNet data, it is referencing ConceptNet 4 unless ConceptNet 5 is explicitly stated.

Appendix. Full Story Text and ConceptNet Justifications

Bullying

Story

Matt and Josh are persons. Matt and Josh are in high school. Matt is insecure because Matt has low self-esteem. Matt bullies Josh. Josh becomes angry. Josh makes a plan because Josh loathes Matt. Josh embarrasses Matt. Matt is sad. Matt stops bothering Josh.

Genesis Rules

- XX and YY are persons.
- XX may bully YY because XX is insecure.
- If XX annoys YY, XX angers YY.
- If XX angers YY, YY may hate XX.
- If XX dislikes YY, XX may harm YY.
- If XX harms YY, YY may stop harming XX.

ConceptNet Justifications

Justification for Matt angers Josh because Matt bullies Josh:

- annoy, bully similar (0.37)
- rule XX angers YY because XX annoys YY.

Justification for Josh embarrasses Matt, probably because Josh loathes Matt:

- harm, embarrass similar (0.57)
- rule XX harms YY, probably because XX dislikes YY.

Justification for Josh angers Matt because Josh embarrasses Matt:

- annoy, embarrass similar (0.72)
- rule XX angers YY because XX annoys YY.

Justification for Matt stops bothering Josh, probably because Josh embarrasses Matt:

- harm, bother similar (0.55)
- harm, embarrass similar (0.57)
- rule YY stops harming XX, probably because XX harms YY.

A Long Day

Story

Fred is a person. John is a person. John is Fred's roommate. Fred works hard at the office. Fred leaves work. Fred buys beer. Fred comes home. Fred opens a beer and watches television. Fred lies down and falls asleep. John comes home. John plays loud music and wakes up Fred. Fred sighs from exasperation. Fred leaves the apartment and takes a walk. The quiet night helps Fred clear his mind. Fred calms down and heads home.

Genesis Rules

- If XX is resting and YY disturbs XX, then XX feels anger towards YY.

ConceptNet Justifications

Justification for Fred wants to relax because Fred works at the office hard:

- work hard CausesDesire relax (0.50)

Justification for Fred buys beer because Fred wants to relax:

- buy beer MotivatedByGoal relax (0.50)
- buy beer UsedFor relax (0.50)

Justification for Fred wants to rest because Fred works at the office hard:

- work CausesDesire rest (1.29)

Justification for Fred relaxes because Fred wants to rest:

- relax MotivatedByGoal rest (0.79)
- relax UsedFor rest (0.50)
- rest HasSubevent relax (1.16)

Justification for Fred wants to take break because Fred works at the office hard:

- work hard CausesDesire take break (0.79)

Justification for Fred rests because Fred wants to take break:

- rest UsedFor take break (0.79)
- take break HasSubevent rest (1.16)

Justification for Fred takes break because Fred wants to rest:

- take break UsedFor rest (0.50)

Justification for Fred takes break because Fred wants to relax:

- take break MotivatedByGoal relax (0.50)

Justification for Fred rests because Fred wants to relax:

- rest MotivatedByGoal relax (0.50)
- rest UsedFor relax (1.40)
- relax HasSubevent rest (0.50)

Justification for Fred relaxes because Fred wants to take break:

- relax UsedFor take break (0.50)
- take break HasSubevent relax (1.00)

Justification for Fred watches television because Fred wants to relax:

- watch television MotivatedByGoal relax (1.00)
- watch television UsedFor relax (0.50)

Justification for Fred watches television because Fred wants to rest:

- watch television UsedFor rest (0.50)
- rest HasSubevent watch television (0.50)

Justification for Fred lies down because Fred wants to rest:

- lie down MotivatedByGoal rest (0.50)
- rest HasSubevent lie down (1.16)

Justification for Fred lies down because Fred wants to relax:

- lie down MotivatedByGoal relax (0.50)
- relax HasSubevent lie down (0.50)

Justification for Fred falls asleep because Fred wants to rest:

- rest HasSubevent fall asleep (1.40)

Justification for Fred falls asleep because Fred wants to relax:

- relax HasSubevent fall asleep (1.00)

Justification for Fred feels anger towards John because John wakes up Fred, and Fred rests:

- disturb, wake up similar (0.34)
- rule XX feels anger towards YY because XX is resting, and YY disturbs XX.

Justification for Fred feels anger towards John because John wakes up Fred, and Fred relaxes:

- disturb, wake up similar (0.34)
- rest, relax similar (0.44)
- rule XX feels anger towards YY because XX is resting, and YY disturbs XX.

Justification for Fred wants to release energy because Fred feels anger towards John:

- anger CausesDesire release energy (0.50)

Justification for Fred sighs from exasperation because Fred wants to release energy:

- release energy HasSubevent sigh (0.50)

Justification for Fred releases energy because Fred wants to relax:

- release energy MotivatedByGoal relax (0.50)

Justification for Fred releases energy because Fred wants to rest:

- release energy MotivatedByGoal rest (0.50)

Justification for Fred wants to cool off because Fred feels anger towards John:

- anger CausesDesire cool off (0.50)

Justification for Fred sighs from exasperation because Fred wants to cool off:

- cool off HasSubevent sigh (0.50)

Justification for Fred cools off because Fred wants to rest:

- cool off UsedFor rest (0.50)

Justification for Fred cools off because Fred wants to relax:

- cool off UsedFor relax (0.50)

Justification for Fred takes a walk because Fred wants to relax:

- take walk UsedFor relax (0.79)

Justification for Fred takes a walk because Fred wants to cool off:

- take walk UsedFor cool off (0.50)

Justification for Fred calms down because Fred wants to cool off:

- cool off HasSubevent calm down (1.00)

Mexican-American War

Story

The United States is a country. Mexico is a country. The year is 1846. Manifest Destiny is popular in the United States. The United States has ambition, has greed, and wants to gain land. Mexico and the United States disagree over borders. The United States move into the disputed territory. Mexican forces attack the United States. The United States declares war on Mexico. Winfield Scott leads the United States army. The United States battles Mexico. The United States triumphs by capturing Mexico City. The United States defeats Mexico and wins the war. The Treaty of Guadalupe Hidalgo officially ends the war.

Genesis Rules

None.

ConceptNet Justifications

Justification for The United States wants to conquer opponent because The United States has greed:

- greed CausesDesire conquer opponent (0.50)

Justification for The United States triumphs by capturing Mexico City because The United States wants to conquer opponent:

- conquer opponent HasSubevent triumph (0.50)

Justification for The United States wants to reach advantage because The United States has ambition:

- ambition CausesDesire reach advantage (0.79)

Justification for The United States conquers opponent because The United States wants to reach advantage:

- reach advantage HasSubevent conquer opponent (0.50)

Justification for The United States conquers opponent because The United States wants to gain land:

- conquer opponent UsedFor gain land (0.50)

Justification for The United States wants to get something because The United States wants to gain land:

- want CausesDesire get something (0.79)

Justification for The United States wants to get something because The United States has greed:

- greed CausesDesire get something (0.50)

Justification for The United States wants to conquer opponent because The United States has ambition:

- ambition CausesDesire conquer opponent (0.50)

Justification for The United States wants to conquer nation because The United States has greed:

- greed CausesDesire conquer nation (0.50)

Justification for The United States wins the war because The United States wants to conquer nation:

- conquer nation HasSubevent win war (0.50)

Justification for The United States conquers nation because The United States wants to gain land:

- conquer nation UsedFor gain land (0.50)

A Long Day, Alternate Ending

Story

Fred is a person. John is a person. John is Fred's roommate. Fred works hard at the office. Fred leaves work. Fred buys beer. Fred comes home. Fred opens a beer and watches television. Fred lies down and falls asleep. John comes home. John plays loud music and wakes up Fred. Fred fights with John over his music. Tempers flare. John apologizes and they drink beer together.

Genesis Rules

- If XX is resting and YY disturbs XX, then XX feels anger towards YY.

ConceptNet Justifications

- work hard CausesDesire relax (0.50)

Justification for Fred buys beer because Fred wants to relax:

- buy beer MotivatedByGoal relax (0.50)
- buy beer UsedFor relax (0.50)

Justification for Fred wants to rest because Fred works at the office hard:

- work CausesDesire rest (1.29)

Justification for Fred relaxes because Fred wants to rest:

- relax MotivatedByGoal rest (0.79)
- relax UsedFor rest (0.50)
- rest HasSubevent relax (1.16)

Justification for Fred wants to take break because Fred works at the office hard:

- work hard CausesDesire take break (0.79)

Justification for Fred rests because Fred wants to take break:

- rest UsedFor take break (0.79)
- take break HasSubevent rest (1.16)

Justification for Fred takes break because Fred wants to relax:

- take break MotivatedByGoal relax (0.50)

Justification for Fred takes break because Fred wants to rest:

- take break UsedFor rest (0.50)

Justification for Fred rests because Fred wants to relax:

- rest MotivatedByGoal relax (0.50)
- rest UsedFor relax (1.40)
- relax HasSubevent rest (0.50)

Justification for Fred relaxes because Fred wants to take break:

- relax UsedFor take break (0.50)
- take break HasSubevent relax (1.00)

Justification for Fred watches television because Fred wants to relax:

- watch television MotivatedByGoal relax (1.00)
- watch television UsedFor relax (0.50)

Justification for Fred watches television because Fred wants to rest:

- watch television UsedFor rest (0.50)
- rest HasSubevent watch television (0.50)

Justification for Fred lies down because Fred wants to relax:

- lie down MotivatedByGoal relax (0.50)
- relax HasSubevent lie down (0.50)

Justification for Fred lies down because Fred wants to rest:

- lie down MotivatedByGoal rest (0.50)
- rest HasSubevent lie down (1.16)

Justification for Fred falls asleep because Fred wants to relax:

- relax HasSubevent fall asleep (1.00)

Justification for Fred falls asleep because Fred wants to rest:

- rest HasSubevent fall asleep (1.40)

Justification for Fred feels anger towards John because John wakes up Fred, and Fred rests:

- disturb, wake up similar (0.34)
- rule XX feels anger towards YY because XX is resting, and YY disturbs XX.

Justification for Fred feels anger towards John because John wakes up Fred, and Fred relaxes:

- disturb, wake up similar (0.34)
- rest, relax similar (0.44)
- rule XX feels anger towards YY because XX is resting, and YY disturbs XX.

Justification for Fred wants to advance battle because Fred feels anger towards John:

- anger CausesDesire advance battle (0.50)

Justification for Fred fights with John over his loud music because Fred wants to advance battle:

- fight MotivatedByGoal advance battle (0.50)

Justification for Fred wants to destroy enemy because Fred feels anger towards John:

- anger CausesDesire destroy enemy (0.50)

Justification for Fred advances battle because Fred wants to destroy enemy:

- advance battle MotivatedByGoal destroy enemy (0.50)

Justification for Fred wants to fight because Fred feels anger towards John:

- anger CausesDesire fight (0.50)

Justification for Fred advances battle because Fred wants to fight:

- advance battle MotivatedByGoal fight (0.50)
- advance battle UsedFor fight (1.00)

Justification for Fred fights because Fred wants to advance battle:

- fight MotivatedByGoal advance battle (0.50)

Justification for Fred wants to die because Fred feels anger towards John:

- anger CausesDesire die (0.50)

Justification for Fred advances battle because Fred wants to die:

- advance battle UsedFor die (0.50)

Justification for Fred wants to release energy because Fred feels anger towards John:

- anger CausesDesire release energy (0.50)

Justification for Fred dies because Fred wants to release energy:

- release energy HasSubevent die (0.50)

Justification for Fred releases energy because Fred wants to rest:

- release energy MotivatedByGoal rest (0.50)

Justification for Fred releases energy because Fred wants to relax:

- release energy MotivatedByGoal relax (0.50)

Justification for Fred wants to kill because Fred feels anger towards John:

- anger CausesDesire kill (0.79)

Justification for Fred dies because Fred wants to kill:

- kill HasSubevent die (1.85)

Justification for Fred kills because Fred wants to die:

- kill MotivatedByGoal die (0.50)

Justification for Fred kills because Fred wants to advance battle:

- advance battle HasSubevent kill (0.50)

Justification for Fred wants to join army because Fred feels anger towards John:

- anger CausesDesire join army (0.50)

Justification for Fred kills because Fred wants to join army:

- join army HasSubevent kill (0.50)

Justification for Fred joins army because Fred wants to fight:

- join army MotivatedByGoal fight (0.50)

Justification for Fred wants to kill person because Fred feels anger towards John:

- anger CausesDesire kill person (0.79)

Justification for Fred joins army because Fred wants to kill person:

- join army UsedFor kill person (1.00)

Justification for Fred kills person because Fred wants to die:

- kill person MotivatedByGoal die (0.79)

Justification for Fred kills person because Fred wants to destroy enemy:

- destroy enemy HasSubevent kill person (0.79)

Justification for Fred kills because Fred wants to destroy enemy:

- kill MotivatedByGoal destroy enemy (0.50)
- destroy enemy HasSubevent kill (0.79)

Justification for Fred kills because Fred wants to kill person:

- kill person HasSubevent kill (0.79)

Justification for Fred dies because Fred wants to relax:

- relax HasSubevent die (0.50)

Justification for Fred dies because Fred wants to destroy enemy:

- destroy enemy HasSubevent die (0.79)

Justification for Fred advances battle because Fred wants to kill person:

- advance battle UsedFor kill person (0.50)