LEARNING KNOWLEDGE TO SUPPORT DOMAIN-INDEPENDENT NARRATIVE INTELLIGENCE

A Thesis
Presented to
The Academic Faculty

by

Boyang Li

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
May 2015

LEARNING KNOWLEDGE TO SUPPORT DOMAIN-INDEPENDENT NARRATIVE INTELLIGENCE

Approved by:

Associate Professor Mark O. Riedl,
Advisor
School of Interactive Computing
Georgia Institute of Technology

Assistant Professor Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology

Professor Ashok Goel
School of Interactive Computing
Georgia Institute of Technology

Assistant Professor Brian Magerko
School of Literature, Media, and Communication
Georgia Institute of Technology

Associate Professor Stacy Marsella
College of Computer and Information Science
Northeastern University

Date Approved: November 14 2014

ACKNOWLEDGEMENTS

Behind every successful dissertation and its author, there is a group of wise people. First and foremost, I would like to thank my advisor Mark Riedl, who I have been fortunate enough to work with since the beginning of my Ph.D. program. Without his guidance, encouragement and support, this dissertation would never have been completed. I must also thank every member of my dissertation committee, whose expertise and experience in multiple research fields have been tremendously beneficial. I am really glad to have had the opportunity to learn from each of you.

Many friends and colleagues have helped me in the writing of this dissertation and other research projects I carried out at Georgia Tech. Many thanks go to Stephen Lee-Urban, Mohini Thakkar, Yijie Wang, Yangfeng Ji, Brian O'Neill, Alexander Zook, Hong Yu, Nicholas Davis, Rania Hodhod and George Johnston.

I benefited from the advice from and discussions with many good friends and great teachers in the research community. For this, I would like to thank Ian Horswill, Emmett Tomai, Jonathan Rowe, Fox Harrell, Wei Lu, Arnav Jhala, David Roberts, and Leslie Kaelbling.

Last but certainly not the least, I must thank my parents and family for their love and support, whom I did not have a lot of time to visit while working on this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Narrative Intelligence is the ability to craft, tell, understand, and respond appropriately to narratives. It has been proposed as a vital component of machines aiming to understand human activities or to communicate effectively with humans. However, most existing systems purported to demonstrate Narrative Intelligence rely on manually authored knowledge structures that require extensive expert labor. These systems are constrained to operate in a few domains where knowledge has been provided.

This dissertation investigates the learning of knowledge structures to support Narrative Intelligence in any domain. I propose and build a system that, from an corpus of simple exemplar stories, learns complex knowledge structures that subsequently enable the creation, telling, and understanding of narratives. The knowledge representation balances the complexity of learning and the richness of narrative applications, so that we can (1) learn the knowledge robustly in the presence of noise, (2) generate a large variety of highly coherent stories, (3) tell them in recognizably different narration styles and (4) understand stories efficiently. The accuracy and effectiveness of the system have been verified by a series of user studies and computational experiments.

As a result, the system is able to demonstrate Narrative Intelligence in any domain where we can collect a small number of exemplar stories. This dissertation is the first step toward scaling computational narrative intelligence to meet the challenges of the real world.

CHAPTER I

INTRODUCTION

> *Our dreams and stories may contain implicit aspects of our lives even without our awareness.*
>
> — Daniel J. Siegel

The long history of narratives in human civilizations can be at least traced back to early myths found in diverse geographical regions, which are believed to define collective identities and justify social institutions [95, 104]. Indeed, narratives in various forms play multiple important functions in human cultures, both for the cognition of individuals and the collective cognition of a society. As a result, *Narrative Intelligence* (NI), or the ability to create, understand and respond to narratives, is believed to be crucial for human intelligence, and by extension, crucial for the simulation of human intelligence [20, 40, 57, 113, 203].

In the past few decades, attempts to computationally simulate Narrative Intelligence has been limited by the reliance on manually authored knowledge, whose cost of authoring limits the scalability of computational systems. In this dissertation, I investigate the problem of learning knowledge in support of Narrative Intelligence tasks. My approach learns narrative knowledge from a corpus of simple stories and applies the learned knowledge to accomplish story generation, storytelling, and story understanding.

This introductory chapter defines Narrative Intelligence and motivates the problem of learning knowledge in support of NI tasks.

1.1 Narratives and Narratology

Before discussing Narrative Intelligence, it is necessary to first define the notion of a narrative. Prince [141] contrasted several definitions of narrative. Here I take a broad definition:

Definition 1 (Narrative)**.** *A narrative is two or more conceptually and temporally related events, which are presented through a particular medium to its audience.*

In this definition, narratives include "novels and romances, novellas and short stories, history, biography and autobiography, epics, myths, folktales, legends and ballads, news reports, spontaneous accounts in ordinary conversation, and so on" [141]. A narrative may be told orally, written in text, performed on stage, presented as audio and video clips, or communicated through other media. Throughout this dissertation, I use the word "story" interchangeably with the word "narrative".

The prototype theory of categorization [157] argues that a concept is defined not by clear-cut rules but by prototypical examples. In this sense, prototypical examples of narrative would include Homer's *Iliad*, Shakespeare's *Hamlet*, and CBS's *The Big Bang Theory*. We could list features for prototypical narratives, such as involving the same human or anthropomorphic characters, consisting of a beginning, a middle and an ending, containing events related by causal relations, etc. Having these features would make a narrative more prototypical and more easily recognizable as a narrative, but in my definition these features are not strictly necessary.

In the study of narratives, the structuralist school has proposed models for breaking down a narrative into multiple levels. In particular, Mieke Bal [5] proposed a three-tier model, the three tiers being denoted as the *fabula*, the *sjuzhet*[1], and the *media* in this dissertation. The fabula includes every event that actually happened to and around characters in the narrative, which may happen simultaneously. The

[1] Alternative spellings include sjužet, syuzhet, suzhet, suzet, sujet, etc.

sjuzhet includes the events that appear in the narrative. That is, the sjuzhet is a subset of the events in the fabula, which have been rearranged into a linear sequence. The ordering of events in the sjuzhet may differ from their ordering in the fabula. Finally, the sjuzhet is realized in the media tier as oral narration, text, video, etc.

Bal's model is not meant to be a cognitive account about how humans create or process narratives, but is used to an analytical tool to delineate the complexity of a narrative and show different layers of construction. I will return to the three tiers when defining Narrative Intelligence. For more details on narratives and tiered narrative structures, see Section 2.1.

1.2 Narrative Intelligence

Cognitive research shows that narratives play important roles in human cognition and communication. Narratives help us communicate complex situations and understand intentions of other people [21, 135]. Storytelling can persuade and impart values [74], create emotional bonding between teller and listener [45], help the construction of self identities [167], and facilitate language learning among children [83].

Given the important role of narratives in human culture and human communication, many AI researchers and cognitive scientists (e.g. [20, 40, 57, 113, 203]) believe that Narrative Intelligence[2] or the ability to craft, tell, understand and respond appropriately to narratives, is crucial for human intelligence, and by extension, crucial for the simulation of human intelligence. Thus, research into Narrative Intelligence should bear significance in the quest for human-level AI and AIs capable of communicating with humans.

Definition 2 (Narrative Intelligence)**.** *Narrative Intelligence is the ability to craft, tell, understand and respond appropriately to narratives.*

[2] The term Narrative Intelligence is said to have been coined at MIT in 1990 by Marc Davis and Michael Travers [41]

Recent years have seen the emergence of real-world applications of Narrative Intelligence. As consumers often consume digital content, such as video games and virtual training scenarios, faster than the content can be produced, automated narrative generation and rendering techniques may help to satiate the appetite for new content. We may, for example, generate new plotlines for games to adapt to player preferences and improve replayability [97]. Similarly, we can create new training scenarios catering to different individual needs [123] and automatically produce cinematographic rendering of 3D scenes [151]. Interactive narratives have been deployed in a number of science, archaeology, and theatrical museums (e.g. [159, 170, 194]). A storytelling robot receptionist was installed at Carnegie Mellon University [70]. Story understanding systems have been used to retrieve cyberbulling stories similar to a user's story [102], to retrieve local viewpoints for news events [107], and to retrieve previous baseball games to be used as commentary [93].

In order to understand the complex information-processing ability that is Narrative Intelligence, it is helpful to study NI as several interrelated components. I propose NI contains a generative aspect and a comprehension aspect. Borrowing from Bal's three-tiered model discussed in Section 1.1, we can further break down both aspects into several components. First, generative Narrative Intelligence can be broken down into the following major components:

[G1] The ability to create a number of events that actually happen in the narrative world, so as to achieve certain aesthetic or communicative goals, such as coherence, suspense or persuasion.

[G2] The ability to create the sjuzhet based on the fabula, so as to achieve certain aesthetic or communicative goals. Common techniques used to create the sjuzhets include, but are not limited to, selecting events from the fabula, rearranging selected events, and projecting the events to a particular point of view (e.g. a personal perspective from a story character).

[G3] The ability to describe or perform the sjuzhet in an acceptable format, such as texts, movies, comic strips, or theatrical performances, to the audience, so as to achieve certain aesthetic or communicative goals. This process creates the media.

In other words, the first three components constitute the process of creating the fabula, transforming the fabula to the sjuzhet, and transforming the sjuzhet to the media. The comprehension aspect of Narrative Intelligence can be similarly broken down to include:

[U1] The ability to infer what events are described by the media. This ability infers the sjuzhet from the media, and is the reverse process of G3.

[U2] The ability to infer what events have happened from the events being described. This ability infers the fabula from the sjuzhet, and is the reverse process of G2.

[U3] The ability to understand the aesthetic and communicative purposes of the narrative and produce appropriate affective responses. Possible responses include common affects, such as happiness or disgust, and narrative-specific affects, such as identification with characters or suspense (cf. [47, 112, 126]).

Thus, narrative comprehension involves backward inference. By reading the text or watching the movie, we are supposed to infer the sjuzhet from the media, and the fabula from the sjuzhet, and interpret their implications. Such backward inference will inevitably encounter uncertainty, so we must allow it to make reasonable mistakes.

Despite being listed separately, the generative and comprehensive aspects of Narrative Intelligence do not work in isolation for the following reasons. First, the two aspects share knowledge structures and operations on the knowledge structures. Second, narrative creation requires the help of narrative understanding. Sharples [163] propose a cognitive theory that creative writing alternates between an engagement stage of intensive writing and a reflection stage where the results are reevaluated. That is, a good story writer probably needs good comprehension skills, so that she can continuously evaluate her own work, predict the audience's affective responses, and revise accordingly. Similar models are proposed by Flower and Hayes [58] and Gervás and León [68]. Finally, story understanding also requires generative abilities, as understanding often requires drawing connections between elements presented separately. The ISSAC story understanding system [120], for example, is capable of creating novel analogies in order to understand novel concepts in science fictions.

I do not claim that the above component-based characterization of Narrative Intelligence accurately captures how the human brain creates, understands or responds to narratives. The main purpose of this description is computational. It provides one possible division of labor for the complex processes underlying narrative creation and comprehension. It may help AI researchers create modular computational systems, or focus on one ability without losing sight of other challenges. Several computational NI systems [4, 33, 119] adopted the division from Bal's three-tier model in order to create a pipelined process. In this dissertation, I implement components G1, G2, G3, and U2 (See Section 1.4).

1.3 Knowledge Intensity of Narrative Intelligence

How can we build computational systems that can demonstrate Narrative Intelligence as characterized above? Evidence from both cognitive science and artificial intelligence systems suggests that in order to achieve Narrative Intelligence, we need a great amount of knowledge about the world and people described in the narratives.

Cognitive studies suggest that in achieving Narrative Intelligence, humans make use of extensive knowledge (e.g. [16, 202]) , which often needs to be acquired and developed over an extended period of time (e.g. [2, 193]). Hudson and Shapiro [81] listed four types of knowledge needed for narrative intelligence: (a) *content knowledge*, or knowledge about events, event sequences, characters and social interactions, (b) *structural knowledge*, or knowledge about the order and manner in which a story is narrated, (c) *microlinguistic knowledge*, or syntactic knowledge for the understanding and production of textual narrative representation, and (d) *contextual knowledge* about pragmatic functions that stories serve for tellers and listeners. The categorization of knowledge by Hudson and Shapiro is consistent with different components of Narrative Intelligence defined earlier. The first three types of knowledge roughly correspond to the creation and understanding of fabula, sjuzhet, and media respectively. The last type of knowledge corresponds to the ability to understand storytellers' intentions and audience's responses (U3).

Existing computational NI systems are also heavily reliant on knowledge. From the knowledge perspective, there are mainly two approaches toward computational Narrative Intelligence: case-based reasoning (CBR) systems and planners. The *case-based reasoning* approach (e.g. [38, 200, 190, 67]) builds Narrative Intelligence on known stories. A new story can be understood in the context of old stories, and snippets of known stories can be combined to create new stories. As one of the earliest research into narrative understanding, Schank and Abelson [161] proposed scripts, or the knowledge of typical events organized in typical temporal and causal relations, as central to the understanding of narratives. Most CBR systems adopt some variants of script. A second type of narrative intelligence systems (e.g. [115, 92, 154] employ knowledge of individual actions with preconditions and effects. By matching preconditions with effects, a *story planner* can combine actions in any causally consistent way to generate stories it has not known before. Notably, some hybrid systems [136, 153, 97] utilize both script-like knowledge and knowledge about individual actions (See Section 2.2 for a more complete review of computational NI systems).

Regardless of the exact representation of knowledge, existing NI systems mostly rely on human experts to manually author the knowledge content, which is usually an expensive process. Although learning is a major component in the 4-phase CBR cycle, most existing CBR systems require complex knowledge that are difficult to learn, so the learning phase is usually ignored. Consequently, most computational systems purported to demonstrate Narrative Intelligence are restricted to micro-worlds where background knowledge has been provided *a priori*. These systems can generate and understand narratives well in a few domains, but their Narrative Intelligence diminishes once the domain of application changes. This knowledge bottleneck has been widely recognized. Nevertheless, the problem of automatically acquiring necessary knowledge to support NI has not been thoroughly investigated.

1.4 Open Narrative Intelligence

In order to tackle the knowledge bottleneck that has troubled computational NI systems, Open Narrative Intelligence systems have begun to attract research interest (e.g. [177, 166, 114]). Open NI Systems are not restricted to a predefined set of domains because they are capable of learning knowledge necessary for Narrative Intelligence in unknown domains.

In this dissertation, I aim to systematically tackle the knowledge bottleneck issue by developing an Open NI system that supports story generation, telling, and understanding. More specifically, it attempts to answer the following research questions:

- How can a computational system acquire structured knowledge in support of narrative intelligence?
- What is a good computational representation for such knowledge?
- How can the learned knowledge be utilized efficiently in story creation, storytelling, and story understanding?

The answer to these questions is summarized as the thesis statement of this Ph.D. dissertation:

> A computational system can acquire knowledge in the form of plot graphs, which accurately describe common situations in the real world, to support Narrative Intelligence in a broad range of domains. The learned plot graphs can support the generation of a large variety of coherent narratives, the telling of narratives in distinct narrator styles, and efficient comprehension of narratives.

This dissertation will demonstrate a system that, from an input corpus of simple exemplar stories, learn complex knowledge structures that subsequently enable the creation, telling, and understanding of stories. The proposed knowledge representation aims to strike a balance between the complexity of learning and the richness of narrative applications, so that we can accomplish the following goals:

- Learn the knowledge robustly in the presence of noise.

- By utilizing the learned knowledge, generate a large variety of coherent fabula that can reasonably happen in the real world. This corresponds to the first generative NI component, G1.

- By utilizing the learned knowledge, select events from the fabula to generate sjuzhet, which corresponds to the NI component G2.

- By utilizing the learned knowledge, tell the sjuzhet with natural language, producing interesting stories. This corresponds to the NI component G3.

- By utilizing the learned knowledge, efficiently infer fabula from a given sjuzhet. This corresponds to the comprehensive NI component U2. and interesting stories based on the learned representation.

The realization of NI capabilities G1, G2, G3, and U2 shows that the learned knowledge representation can effectively and efficiently support computational Narrative Intelligence, in both story generation and story understanding. To the best of my knowledge, this is the first computational system capable of learning knowledge to support both aspects of NI. I implement all story generation capabilities but not all story understanding capabilities. The capability U2 is selected because inferring fabula from sjuzhet is the most relevant to the plot graph representation, which for the most part represents story structures in terms of events and their interrelationships. Implementing U2 hence helps us to examine if story understanding can be efficiently performed on this representation.

As a result, my system will be able to demonstrate NI in any domains where a small number of exemplar stories can be collected. The collection process is easy because writing the stories does not require training in computer science. One inexpensive way (but not the only way), as I demonstrate, is to crowdsource the exemplar stories from Amazon Mechanical Turk.

In addition to long-term scientific contributions, the development of the SCHEHERAZADE system brings about immediate benefits in terms of real-world applications. The system's capability to learn socio-cultural conventions can be employed for building virtual training scenarios that familiarize users with foreign cultures or guide children with autism to adhere to social conventions. It could be expensive to hire experts to author materials to cover many social situations in a foreign culture, such as interviewing for a job in Japan or greeting the locals in Samoa. In comparison, the SCHEHERAZADE system only requires a small number of exemplar stories written by English-speaking non-experts who have some experience with the foreign culture and that particular social situation. It was shown that crowdsourced data can help the creation of intelligent systems help people with autism to learn about social situations [14]. Moreover, the SCHEHERAZADE system can provide background stories and diverse narration styles to support believable virtual characters who talk about their daily activities in the virtual world. Believable virtual characters find applications in games, virtual training environments and healthcare. Section 6.3 contains a more detailed discussion on the applications of SCHEHERAZADE.

1.4.1 Necessity of Script-Like Knowledge

As discussed earlier, current Narrative Intelligence systems make use of two major forms of knowledge: scripts and action templates, which brings up the question which knowledge representation should we learn. While both types of knowledge are useful, I choose to learn script-like knowledge for the reasons explained below.

Despite the flexibility of action templates, I argue that action templates by themselves are not sufficient for the task of generating and understanding complex narratives. There are two major reasons: First, reasoning about everything from first principles is computationally slow. Even simple daily situations such as buying medicine from a pharmacy store can involve complex reasoning: the pharmacist performs information-seeking actions to check if the customer has the necessary prescription; the customer requests the receipt as a contingency plan in case the medicine needs to be returned; both parties may need to reason about the intentions of the other. Although each problem can be solved computationally (e.g. with techniques discussed in [52, 96, 152]), solving them every time is not as efficient as executing a known script. Second, in many social situations, we need to respect social conventions, which may not have resulted from clear rational reasoning at the present time. For example, the choice of handshaking versus bowing as greetings may have important historic reasons (such as hygiene conditions), but those reasons may not be present at our time. Thus, respecting social conventions requires us to follow existing scripts.

My approach makes use of a script-like representation called a *plot graph*. A plot graph representation contains vertices and edges. Vertices represent events that can occur in the given situations. Some events are optional, indicating that their occurrence is not required for the legality of the event sequence. There are two types of edges in a plot graph. A unidirectional precedence relation signifies that one event strictly occurs before another event. The precedence relations form a partial-order model, which allows for variations in legal event sequences for the situation. Precedence relations coincide with causal and temporal precedence information, which are important for narrative comprehension (cf. [56, 72, 187]). A bidirectional mutual exclusion relation between two events signifies that the two events cannot occur together in any story. Mutual exclusion relations provide support for situational variations and alternatives.

An example plot graph portraying a pharmacy situation is shown in Figure 1. The unidirectional precedence relations are shown as arrows, and the bidirectional mutual exclusion relations are shown as dashed lines. The graph also includes an optional event "Customer takes change". A formal definition of this representation can be found in Section 3.1.

1.4.2 The Scheherazade System

This section provides an overview of the Scheherazade system[3], and lays out the organization of this dissertation.

The Scheherazade system learns the structure of events in a given situation from crowdsourced exemplar stories describing that situation. The systems acquires a number of simple exemplar stories for a particular social or procedural situation from Amazon Mechanical Turk (AMT).

After a corpus of stories are acquired, the learning of the plot graph proceeds in four steps. First, we cluster sentences with similar semantic meaning from exemplar stories, each cluster becoming one event in the plot graph. In order to reduce the

[3] Different from the story annotation scheme by David Elson [50] with the same name.

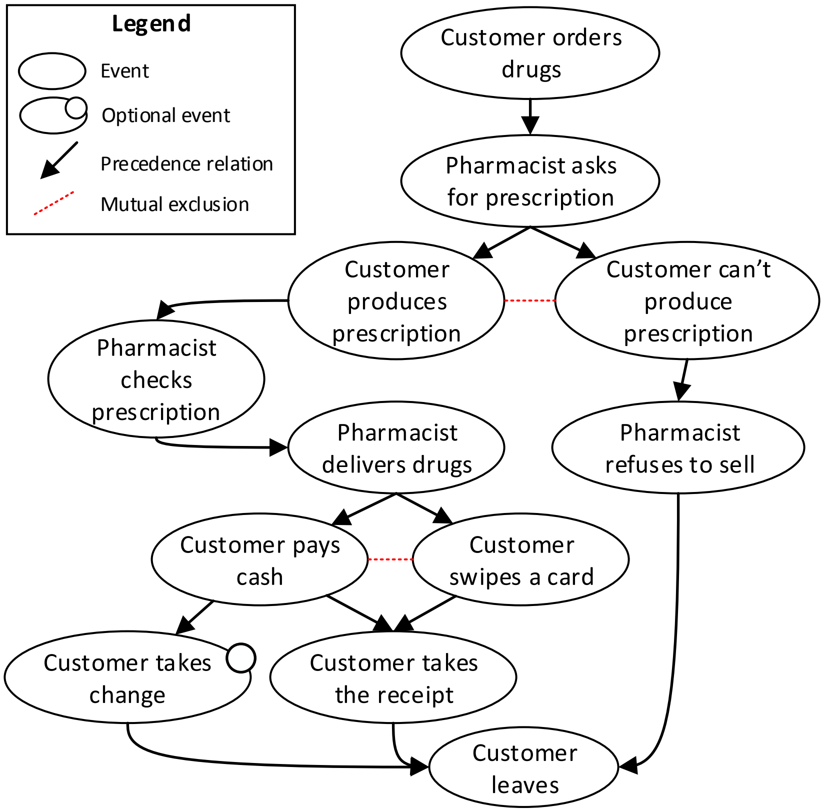

Legend
Event
Optional event
Precedence relation
Mutual exclusion
Customer orders drugs
Pharmacist asks for prescription
Customer produces prescription
Customer can’t produce prescription
Pharmacist checks prescription
Pharmacist delivers drugs
Pharmacist refuses to sell
Customer pays cash
Customer swipes a card
Customer takes change
Customer takes the receipt
Customer leaves

Figure 1: A sample plot graph for the pharmacy situation.

difficulty in natural language processing, we have asked crowd workers from AMT to use simple language, i.e., using one sentence with a single verb to describe one event, avoiding pronouns, etc. In the second step, we identify precedence relations between the events. The third step learns mutual exclusion relations using mutual information. The final step identifies certain events as optional. Chapter 3 describes the learning of plot graph and the evaluation of learned graphs.

Story generation in SCHEHERAZADE is the process of generating a linear sequence of events while respecting the constraints posed by the precedence relations, mutual exclusion relations, and event optionality. This linear sequence contains all events that are presumed to have happened in the virtual world, which can be seen as a linearized fabula. SCHEHERAZADE further selects a subset of fabula events in order to create an interesting sjuzhet, and transform the sjuzhet into a textual description. Chapter 4 details the generation of stories from the learned representation and their evaluation.

Chapter 5 tackles the problem of efficient story understanding. When we infer fabula from sjuzhet, it may be computationally expensive to estimate the probability of a certain event happening in the fabula. Although the general problem is shown to be NP-hard, I present methods to reduce its computational cost for commonly encountered plot graphs. Finally, Chapter 6 summarizes and concludes this dissertation.

CHAPTER II

RELATED WORK

> *What scientists do when they look at the line of bubbles on the screen is work out the story of the particle that made them: what sort of particle it must have been, and what caused it to move in that way, and how long it was likely to continue.*
>
> — Philip Pullman

This chapter starts by reviewing theories of narrative and narrative structures to build a theoretical background and introduce vocabulary for subsequent discussions. After that, I review traditional Narrative Intelligence systems based on manually coded knowledge, as well as existing work for learning script-like information. Combining the strengths of those work, Open Narrative Intelligence systems learn knowledge in order to support Narrative Intelligence. These systems are discussed and compared in the last section.

2.1 Narratives and Narrative Structures

On the definition of narratives, there are two major schools of thoughts. In classic narrative theory (e.g. [61, 171]), a narrative is defined as the opposite of drama. A narrative is communicated by a narrator whereas a drama presents the actions and scenes directly. This school of thought emphasizes this *mediacy* as the most important feature of narrative.

Despite the apparent differences, from an information-processing perspective the understanding of an oral narrative, a theatrical drama, and a motion picture likely share some common processes. For example, there is likely a process to reconstruct a timeline of events, as events may have been presented out of order and simultaneous events have been fit into a linear sequence. There is likely a process that infers events that have happened but not presented in order to understand the events that are presented. Although the drama and the motion picture seem more direct than the oral narrative, they are not identical to the real world. The theatrical stage, for example, may represent an apartment, a garden, or even a battlefield. The audience usually have to understand certain stage and cinematographic conventions in order to understand dramas or movies (cf. [11, 109]).

AI Researchers looking for an information-processing account of narratives turn to the structuralist school (e.g. [164, 182, 185]), which defines narratives around events that make change to a fictional world and relate to one another. Structuralists treat oral narratives, novels, dramas, and movies all as legitimate narratives. Nevertheless, it can be difficult to establish a detailed consensus definition for narrative, as structuralists' definitions differ in basic features such as the minimum number of events required and if causal relationships are necessary in additional to temporal relationships. E. M. Forster [59] famously claimed that "the king died and then the queen died" is a story, whereas "the king died, and then the queen died of grief" qualifies as a plot for it introduces a causal relationship. To Forster, a story needs at least two events, and the plot presented to the audience should contain causal relationships (see Section 2.1.1 for the difference between story and plot). Genette [64] argued that a single event "the king died" can qualify as a minimal story. Tomashevsky [185] believed causality is necessary for stories. In contrast, Schmid [162] considers the difficulty of unambiguously determining if causal relations exist in a story or any forms of text, and argues temporal relations should be sufficient for the definition of stories. The interested reader is referred to Schmid [162] and Prince [141] for further details on this subject.

Aiming at achieving Narrative Intelligence computationally, I attempt to avoid this debate by taking a broad and practical definition:

Definition 1 (Narrative)**.** *A narrative is the telling of two or more conceptually and temporally related events, which are presented through a particular medium to an audience.*

Further clarification is required for this definition. The medium of a story can take many forms, such as text, comic, audio or video. By this definition, narratives include “novels and romances, novellas and short stories, history, biography and autobiography, epics, myths, folktales, legends and ballads, news reports, spontaneous accounts in ordinary conversation, and so on” [141]. The telling of a narrative requires a narrator and one or more narratees, which may be made explicit or implicit. A narrative may also be told to oneself, either spoken out loud or silently. I exclude a story with one event to distinguish a story from an event description. After all, a story with only one event is uncommon in most daily scenarios. This dissertation treats the words story and narrative as synonyms.

The prototype theory of categorization [157] argues that a concept is defined not by clear-cut rules but by prototypical examples. In this sense, prototypical examples of narrative would include Homer’s *Iliad*, Shakespeare’s *Hamlet*, and CBS’s *The Big Bang Theory.* We could list features for prototypical narratives, such as involving the same human or anthropomorphic characters, consisting of a beginning, a middle and an ending, containing events related by causal relations, etc. Having these features would make a narrative more prototypical and more easily recognizable as a narrative, but in my definition these features are not strictly necessary.

2.1.1 Tiered Models of Narratives

In the structuralist tradition, we can further break down a narrative into a tiered generative model of narrative. The model is not meant to be an accurate and plausible cognitive model about how stories are actually generated by human writers, but a tool for analytical purposes. Structuralists such as Tomashevsky [185] and Bal [5] used this model as a way to analyze narratives and reader's understanding of narratives. Such models also enable Artificial Intelligence programs to generate stories in a pipelined fashion (e.g. [4, 33, 150]).

The idea that a narrative can be analyzed in terms of two tiers, a *fabula* and a *sjuzhet*, can be traced back to theorists earlier than Tomashevsky. However, Tomashevsky [184, 185] probably provided the earliest clear statement of their differences and purposes. The fabula refers to the events that happen in the narrative-depicted world. The sjuzhet refers to the description of events from which a reader learns about the fabula. As various artistic techniques are used to produce the sjuzhet, such as describing events out of order (e.g. flashbacks), describing events from different perspectives, or omitting events, the reader must reconstruct the fabula from the sjuzhet in order to understand the narrative [63]. In English, the fabula is referred to as the story and the sjuzhet is often referred to the plot or discourse, which is not dissimilar to the dichotomy by Forster [59] described earlier. French theorists Todorov [182] and Genette [64] describe similar dichotomy as *histoire* and *discours* or *histoire* and *récit* respectively.

The concept of sjuzhet in the above two-tier model may be still considered ambiguous. Bal [5] proposed a three-tier model that further differentiates between the text in which the events are presented (which she called the *text*) and the events being presented in the text (which she called the *story*). As narrative analysis is not limited to the textual form, in this dissertation, I denote the three layers as fabula, sjuzhet, and media.

Schmid [162] recently proposes a four-tier model. Schmid notes that the real world contains infinite amount of details. Thus, there must be a process where the author selects what constitute an event that is eligible to be included in the fabula. This selection process is different from sjuzhet composition. Concepts of events must be abstracted from artificially segmented perceptual inputs of the human senses.[1]

This dissertation follows the three-tier model proposed by Bal [5]. Thus, I consider a narrative to contain three tiers, as defined below:

Definition 3 (Fabula)**.** *A fabula is the complete set of events that include but are not limited to all events presented in the narrative, that happen to and around all story characters within a continuous time period, and that are temporally partially ordered as multiple events can happen simultaneously.*

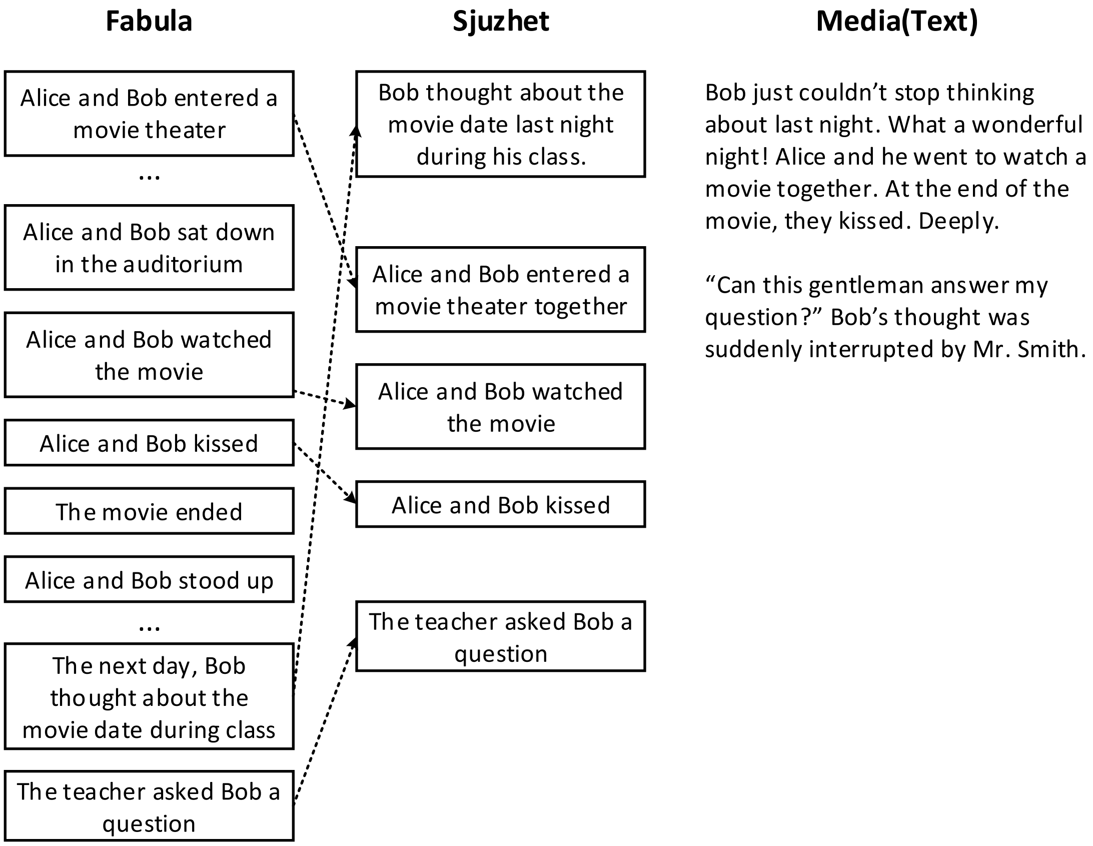
Definition 4 (Sjuzhet)**.** *A sjuzhet is a linear sequence of events that are selected from the fabula to constitute the narrative, possibly in a different ordering from the fabula and with possible repetition of events. One event may be repeated from multiple viewpoints, such as viewpoints of an omniscient narrator or different story characters.*

Definition 5 (Media)**.** *The media is the form of representation in which the sjuzhet is actualized and presented to the audience, including but not limited to oral narration, audio recordings, video footages, theatrical performances, text, multimedia hypertext, and so on.*

An example story that illustrates the differences between the three tiers of fabula, sjuzhet, and media is shown in Figure 2. Many different literary techniques may be used to transform a fabula into a sjuzhet, including but not limited to: *linearization*, which turns partial-ordered events into a single sequence that is suitable to telling; *reorganization*, which orders the events differently from the order in which they actually occur (e.g. flashbacks); *compression* and *expansion*, which selects more or less events from the fabula, creating a slower or faster pace; *focalization*, which tells events from different points of view.

This dissertation follows this three-tier model and proposes methods to generate fabula, sjuzhet, and text respectively. For the purpose of this dissertation, the task

[1] Other complex models have been proposed. See, for example, Branigan's 8-tier model [16], whose structure is vastly different from those reviewed.


Fabula
Sjuzhet
Media(Text)
Alice and Bob entered a movie theater
...
Alice and Bob sat down in the auditorium
Alice and Bob watched the movie
Alice and Bob kissed
The movie ended
Alice and Bob stood up
...
The next day, Bob thought about the movie date during class
The teacher asked Bob a question
Bob thought about the movie date last night during his class.
Alice and Bob entered a movie theater together
Alice and Bob watched the movie
Alice and Bob kissed
The teacher asked Bob a question

Bob just couldn't stop thinking about last night. What a wonderful night! Alice and he went to watch a movie together. At the end of the movie, they kissed. Deeply.

"Can this gentleman answer my question?" Bob's thought was suddenly interrupted by Mr. Smith.

Figure 2: An example story contrasting the fabula, sjuzhet, and textual media

of event segmentation and abstraction, as described by Schmid [162], is performed by human crowd workers from Amazon Mechanical Turk and is not modeled directly by AI.

Although the three-tier model was not proposed as a model of how human writers actually write stories, some reasoning about the fabula, sjuzhet, and media is likely to have happened during writing. For example, to write a detective novel, one probably has to first think of a criminal plot, which is the fabula. The detective usually explains this plot, which is realized as the sjuzhet and the text. For computational systems that aim to generate narratives, this three-tier model provides a useful division of labor and a modular system design. The next section reviews those systems.

2.2 Computational Narrative Intelligence

In this section, I review traditional story generation systems that rely on manually coded knowledge. In line with the definition of Narrative Intelligence in Chapter 1, I categorize generative systems into fabula generation, sjuzhet generation, and text generation. Story understanding systems are categorized into systems that make factual inferences and systems that predict or simulate human readers' responses to narratives.

2.2.1 Fabula Generation

As defined previously, fabula refers to the events that actually happen in the world, not the events presented to the audience. Thus, strictly speaking, one cannot really present a fabula. However, many computational systems do not explicitly reason about or make changes to sjuzhet structures, and can be thought of as employing a simple means of presenting a linear sjuzhet that includes everything in the generated fabula or directly adopting the sjuzhet existing in a retrieved story case. To highlight the fact that these systems do not reason directly about the sjuzhet, I call these fabula generation systems.

There are two major approaches for fabula generation and Narrative Intelligence in general: case-based reasoning and planning. These systems utilize different knowledge structures and different operations on these structures. Some recent systems make use of both case-based reasoning and planning.

Computational case-based reasoning (CBR) [87] is motivated by the observation that humans use past experiences to guide decision making. Hence, Narrative Intelligence can be built on top of a collection of known stories or story snippets. A new story can be understood in the context of old stories, and known story segments can be combined to create new stories. The notion of script was proposed as one of the earliest research into narrative understanding. Minstrel [190] is an elaborate

CBR story generation system that reuses known stories and adapts them using rules (called TRAMs) that recognizes story snippets similar to a query and transform the snippet to satisfy the query. The story structure is constructed by filling in templates. Minstrel utilizes three types of knowledge, including known stories, recipes to achieve story themes and dramatic effects, and methods for adapting stories. All three types of knowledge are manually authored.[2]

Another inspiration of the case-based approach comes from the narratologist studies that found many narratives contain combinations of similar building block. Propp [142] found that the majority of Russian folklores can be explained by the combination of 31 high-level narrative functions, which are semantic units that serve similar purposes in the narrative, such as the hero receiving an interdiction and the hero subsequently disobeying the interdiction. Studies of North American Indian and Eskimo folktales yielded similar results [35, 46]. In the book *Plotto* published in 1928, the prolific novelist William Cook summarized popular plots as numerous interconnected recipes. Gervás *et al.* [67] proposed a case-based reasoning system that retrieves and combines Propp's functions into stories. Characters in the functions can be replaced for coherence and similar functions can be substituted.

The Riu system [129] generates steam-of-consciousness stories. The system represents cases as a temporal sequence of actions organized as a semantic network, where each action is associated with a textual template. The retrieval of cases is performed by matching identical text and structural analogy computed by the Structural Mapping Engine [54]. Similarly, Riedl and León [149] transform a story analogically from one domain to another and uses planning to fix incoherence.

The complete CBR cycle contains four phases: Retrieve, Reuse, Revise, and Retain. The Retrieve phase locates an existing case, which is modified to solve a new

[2] A recent reconstruction of Minstrel [181] can be downloaded from http://minstrel.soe.ucsc.edu

problem in the Reuse phase. The Revise phase checks the newly formed solution against reality. The Retain phase keeps good solutions for future use. Interestingly, most CBR systems for story generation do not have full-fledged Revise and Retain phases. This is probably because it is difficult to validate fictional stories, so it may be undesirable to keep unvalidated stories. The complexity of knowledge used by such systems also makes learning difficult.

If we have a very large source of story cases and organize them effectively in memory, the CBR approach could theoretically work well in many domains. However, due to the lack of the Retain phase, and the cost of manually authoring cases, CBR systems can suffer from the knowledge bottleneck. Due to the lack of cases, we may not find a case that is close enough to the story we wish to generate or understand. When the target story and the retrieved case differ too much, substantial adaptations are required, and too many changes to the case risk making it incoherent. The lack of the Revise phase makes it difficult to detect incoherence, aggregating the problem. As an example, Peŕez y Peŕez and Sharples [137] pointed out Minstrel may replace a girlfriend with a horse because both can be kissed by a knight.

A second type of Narrative Intelligence systems are the *story planners*. Story planners utilize a world model consisting of a number of action templates. An action template contains a number of precondition propositions, which must be true for this action to happen, and a number of effect propositions, which will become true after this action happens. An action template may be lifted, i.e. containing variables to be instantiated. For example, the `eat(?person, ?food)` action contains two variables `?person` and `?food` that can bind to different entities like `Adam`, `Betty`, `noodle` or `pizza`. In order to achieve a given goal state, a planner instantiates lifted actions and links individual actions into a complete sequence, where the preconditions of all actions are satisfied by effects of earlier actions and the initial state.

Causal structures in a plan enable complex reasoning about story structures and aesthetics, so a large number of story planning systems have followed this line of research. The planner could be used to achieved goals of individual story characters, or author goals about how the story should end. Talespin [115] is the first application of planners in story generation and plans the actions for individual characters in the hope that some interesting stories will emerge. Riedl and Young's Intentional Partial-Order Causal-Link Planner (IPOCL) [154] considers both character intentions and author goals. Each character intention creates an intentional frame, and each intentional frame contains actions that achieves that intention and is motivated by another action. This allows IPOCL to create both coherent characters and satisfy authorial intent. The Virtual Storyteller [179] system split each character into two planners with different mindsets. The first one is an in-story character who is not aware of anything outside direct perception, and the second is is an out-of-story actor who is intentionally aiming for achieving story goals. Porteous *et al.* [139] explicitly considered time during story planning. Ware and Young [196] extended IPOCL to handle possible intentional conflicts in multiagent narratives, but the algorithm does not automatically create or avoid conflicts. Brenner [17] proposed narrative generation with multiagent planning and limited look ahead. Si *et al.* [165] used partially observable Markov processes, which might be considered as planning under uncertainty, to model characters in interactive narratives. As an exception to the rule, Cavazza *et al.* [24] presented a Hierarchical Task Network (HTN) planner to create character behaviors. A HTN planner possesses the knowledge of how each higher-level action can decompose into a sequence of lower-level actions. This knowledge is probably more similar to scripts than to action templates.

By breaking stories into small building blocks, the planning approach is capable of generating novel stories while maintaining the story coherence, which is usually defined as characteristics of story plans (e.g. actions being justified by proper intentions [154] or not having dead ends [97]). However, there are two reasons why action templates cannot completely replace script knowledge. First, scripts provide condensed knowledge that can be slow to compute directly from action templates. For example, reasoning about characters' intentions can be very expensive, as every character may be reasoning recursively about other characters' reasoning. Second, scripts do not capture social conventions that are not completely results of logical deductions. Therefore, social conventions need to be encoded as scripts.

As action templates and scripts have different strengths, some recent systems utilize both types of knowledge representation. The Universe system [92] uses decompositions of high-level actions into low-level actions as well as actions' preconditions and effects to guarantee the coherence of the generated story. The planning mechanism in Universe appears to lack backtracking and is a precursor to the modern Decompositional Partial-Order Causal-Link Planner (DPOCL) [205]. Mexica [136] models creating writing according to the model of two alternating modes by Sharples [163]. The engagement mode uses case-based reasoning to project the next major action for characters, whereas the reflection mode uses planning to fill in missing details. Flaiclough and Cunningham [53] used case-based planning with Propp functions to generate stories in a multi-player game. Riedl and Sugandh [153] proposed a system similar to multi-case based planning, which links multiple small story vignettes with planning in order to maintain coherence. Li and Riedl [97] adapt human-authored plotlines in games, which are represented as decompositional partial-order plans. They extend the DPOCL algorithm to remove certain undesirable causal structures, such as actions that do not contribute to the main storyline (i.e. dead ends). The Doraemon gadget generator [99] creates high-tech or magical gadgets that achieve impossible deeds in fictions by analogically modifying the behaviors of common objects which are represented as partial-order plans. The ability to create highly imaginative gadgets illustrates the power of hybrid systems.

Like CBR systems, story planners are also significantly restricted by the availability of manually authored knowledge. Hybrid systems build their strengths from the utilization of both scripts and action templates, but the need to author both types of knowledge aggravates the problem of knowledge bottleneck. Due to the substantial time and financial cost of authoring knowledge, the knowledge available to those systems usually can describe only a few domains. As a result, these NI systems can only operate in these domains. Some of those systems, such as PAM or ISSAC, can produce results almost rivaling human intelligence in a known domain, but they cannot operate outside such domains at all.

The proposed dissertation aims to address this knowledge bottleneck by developing methods to acquire structured knowledge in support of Narrative Intelligence in arbitrary domains. Complex representations are usually powerful but difficult to learn. By adopting the plot graph representation, my approach aims to strike a balance between the complexity of representation and the ease of learning.

2.2.2 Sjuzhet Generation

A few computational systems have focused on sjuzhet techniques. Bae and Young [4] explored the use of sjuzhet techniques, including flashbacks and foreshadowing, based on causal structures in plans. They detected *significant events* that directly contribute to goal states, and *initiating events* that are on causal chains leading to the significant events. An initiating event that is not causally linked to any other event is called separable, as the removal of this event will not affect the rest of the narrative other than the corresponding significant event. This initiating event is then omitted from the narrative, and presented after the significant event using flashback. Foreshadowing could be used to allude to the Initiating Event without completely giving the event away.

Cheong and Young [34] created suspenseful sjuzhet based on a given fabula. They first detected a core set of events, or kernels, without which the story could not be understood. The rest of events are called satellites. Drawing from Trabasso and Sperry [187], events that have a large number of incoming and outgoing causal links and are close to the goal state are recognized as kernels. They followed the proposal of Gerrig and Bernardo [65] and argued that overall suspense increases as the number of plans in which the protagonist achieves her goals decrease, and the number of plans in which the protagonist fails increase. Therefore, the potential suspense of each event is computed from how many of its effects can threaten the protagonist's goal and how much it can support the protagonist's goal. A suspenseful story is created by adding events outside the kernel set with high potential suspense to the kernel events. O'Neill and Riedl [128] provided another interpretation of Gerrig and Bernardo [65] for the detection of suspense, which is reviewed in Section 2.2.4.

Another interesting problem in sjuzhet generation is focalization (cf. [3, 206]), which refers to the technique of storytelling from multiple viewpoints, such as viewpoints of an omniscient narrator or any story characters. In addition to the omniscient foci, storytelling could be told from an external foci, where we know the external behaviors but not the mental activities of a character, or an internal foci, where we know both the external behaviors and the mental activities of a character. Bae *et al.* [3] propose a framework for generating different internal focalization using different libraries of actions templates. Porteous *et al.* [138] use a planning approach where stories under different viewpoints are turned into constraints for planning.

The Curveship system [119] has an elaborate Teller module that supports sjuzhet techniques including reordering the events, changing the frequency, speed, and focalization of narration. The fabula events may be ordered in chronological order, reverse chronological order, or in a zigzag form that interleaves the present and the past. It also support flashback, flashforward, categorical and random ordering of events. The system contains an advanced system of text templates for realizing textual descriptions for events.

Most of the sjuzhet generation systems are based on planners, possibly due to the rich set of information that causal structures of plans can provide. In Section 4.3, I will discuss the detection of how typical an event is to a learned plot graph, and how to use the typicality values to create sjuzhets. The task is similar to the detection and use of kernel and satellite events in [34].

2.2.3 Story Text Generation

There are also research effort on creating the media layer of narratives. The AUTHOR system [22] translates a symbolic plan generated by a story planner into a textual story with several major steps: (1) segmenting the story into sentences and paragraphs, (2) scanning from the beginning to end to determine articles, pronouns, and contextual references, (3) making lexical choices to avoid repetition, and (4) aggregating and reordering clauses to create complex and compound sentences. After that, the intermediate product goes through a surface realizer to produce the final story text.

Based on the Big Five model of personality [188, 124], Mairesse and Walker created the PERSONAGE model [111] that maps the Big Five dimensions (Extraversion, Emotional Stability, Agreeableness, Conscientiousness, Openness to Experience) to a wide variety of language features including semantic content, syntactic structure, sentimental polarity, vocabulary, and pragmatic markers such as “kind of” and “you know”. The computational implementation transforms a symbolic discourse plan with precise semantic meaning into text according to the mapping in order to portray different personality. Although the PERSONAGE model covers broad ground, the content of the discourse plan could limit the generated text. For example, if the system

Table 1: Examples of questions answered by SAM to illustrate its capability of story understanding, produced by SAM after reading a story about John went to a restaurant and received a burnt hamburger. Adapted from [37].

Q: Did John sit down in the restaurant?

A: Probably.

Q: What did the waiter serve John?

A: The witer served John a hamburger.

Q: Why didn't John eat the hamburger?

A: Because the hamburger was overdone.

Q: Did John pay the check?

A: No. John was angry because the hamburger was overdone and so he left the restaurant.

is not aware of possible paraphrases, it may not find the best paraphrase that characterize a given personality. Rishes *et al.* [155] used Personage to create different tellings of stories generated from a semantic representation consisting of events and character intentions [50]. The generated linguistic styles differ mostly in pragmatics rather than content.

Instead of generating from symbolic representations with precise semantic meaning, I generate story text by selecting from existing sentences that are similar but not strictly synonymous to describe an event (i.e. sentences may differ in content). I consider parameters directly related to word choices: degree of details, fictionality, and sentiments. Section 4.4 contains a detailed description.

2.2.4 Story Understanding

The comprehension aspect of Narrative Intelligence, as defined in Chapter 1, includes the ability to (1) infer facts (from text to sjuzhet, and sjuzhet to fabula) and (2) understand purposes of narratives and respond to narratives affectively. Story understanding systems can be categorized according to which of the two abilities they achieve.

The first generation of story understanding systems mainly focus on the factual inference problem. Schank and Abelson [161] proposed that scripts, containing the knowledge of typical events organized in typical temporal and causal relations, are central to the understanding of narratives. Following their lead, a number of systems aim to understand stories by matching stories with known scripts consisting of slots to be filled. SAM [38] applies known scripts to match a given story, so that it can fill in empty slots in the script, resolve coreference, and predict events.[3] PAM [200] extends SAM to make use of plans and causal relations between events to infer characters' intentions. PAM considers the relationships between different characters' intentions, such as conflicts and coordination. Both SAM and PAM use question and answering to demonstrate their story understanding capabilities. Table 1 shows some questions and answers given by SAM in [37]. Recognizing the difficulty of manually authoring scripts that are sufficiently numerous and detailed to handle the complexity of real-world stories, the AQUA system [145] aims to learn from stories to improve scripts. Section 2.5 contains a more detailed review of AQUA.

The ISSAC story understanding system [120] is notable for it transcends the boundary between story generation and story understanding. Utilizing a vast set of knowledge, ISSAC is capable of understanding novel concepts in science fiction by creating analogies. In today's terms, these analogies are conceptual blends [189], such as the blending of robotic manufacturing machines with human intelligence to create a robot commonly seen in science fictions.

Several other work aim to model the human readers' cognitive process and to produce similar responses. Niehaus [122] and Cardona-Rivera *et al.* [156] build situation models for narrative comprehension. These models determine salience of the events in human readers' memory, as they read a narrative sequentially.

[3] In a candid historical account, Wendy Lehnert notes that due to the limited amount of knowledge that SAM possesses, its input has to be "reverse-engineered to stay inside the limitation of SAM's available scripts" [94], although it was only meant to be a prototype.

O'Neill and Riedl [128] designed the Dramatis system for simulating how humans detect suspense in stories. They offer another interpretation of Gerrig and Bernardo [65] that is different from Cheong and Young's [34]. O'Neill and Riedl argue generating all possible plans to see if the number of successful and failed plans increase or decrease is computationally impractical and cognitively implausible. Instead, they model the audience as searching for the most likely plan in which the protagonist avoids a negative outcome, where likelihood is correlated with ease of cognitive retrieval. Thus, a plan that refers to actions that have been activated or primed previously is considered more likely. When the audience find it difficult to think of an avoidance plan, suspense ensures.

The EA NLU system developed by Tomai [183] is mainly concerned with the communicative purposes of narratives. The system can perform three understanding tasks: (1) what is a proper moral decision in the situation described by a narrative, (2) who is to blame for a negative outcome, and (3) what is the moral of a story. EA NLU is based on ResearchCyc [39], which expresses over 5 million facts in higher order logical propositions, but narrative-specific extensions still have to be made at times. Tomai defines a set of narrative functions that readers expect from a story. The natural language processing leverages this knowledge to abduct the meaning of simple sentences.

In addition to the above symbolic approaches for story understanding, recent years have seen the emergence of machine learning approaches toward limited semantic understanding of narratives, such as extraction of characters, events, and emotional trajectories. Bamman *et al.* [6, 7] learn simple character types from novels. Valls-Vargas *et al.* [192] detect heros, villains, and other characters by comparing extracted characters' actions against a predefined character interaction matrix. Following the narrative analysis by Labov and Waletzky [90, 91], who classified several types of narrative clauses, Ouyang and McKeown [132] and Swanson *et al.* [178] detect the major actions in narratives. Mohammad [117] visualize the emotional trajectories of novels using the number of emotional words. Elsner [49] computes similarities between novels using character mentions and character emotions. While being able to capture some structural information, Elsner notes the technique is not yet sophisticated enough to produce role-based narrative summaries.

2.3 Detecting Lexical Sentiments

In Section 4.4, the SCHEHERAZADE system recombines crowdsourced sentences into a natural language presentation of a story. In order to generate stories with positive or negative sentiments, I detect the sentiment of sentences by aggregating sentiments of individual words. In the current section, I briefly review relevant work in the detection of sentiments and emotions in natural language text. Sentiment analysis has gathered significant research interests over the past few decades (e.g. [133, 169]. This review only concerns work that detect sentiments and emotions of individual words.

Work in this area start with having linguistic experts annotating the emotions associated with English words (e.g. [172]). Similar to the situation of narrative knowledge, expert annotations for a large number of words are accurate but expensive to acquire. Thus, there is generally a trade-off between the amount of annotation and their accuracy, leading to a spectrum of dictionaries with varying degrees of coverage and accuracy.

Expert annotations are usually the most accurate type of emotional dictionaries but usually are limited to a small number of words and positive/negative polarity. The General Inquirer lexicon [172] contains 11788 words where 1915 are labeled as positive and 2291 are labeled as negative. It also contains additional sentiments categories include hostile, strong, weak, etc. Cerini *et al.* [25] developed the Micro-WNOp corpus[4] containing 1105 WordNet synsets annotated with their positivity and negativity with real numbers in the [0,1] interval. As an extension of WordNet [116], where word senses are organized as synsets, WordNet-Affect [173, 174] systematically selected words and labeled if the word describes emotion, mood, cognitive state, emotional response and other aspects of affects. When a word is labeled as an emotional word, it is further classified into positive, negative, ambiguous and neutral. WordNet-Affect contains 1637 words and 918 synsets that are labeled as emotional.

The emerging technique of crowdsourcing also provides an cost-effective way to annotate words' emotions. Approaches such as the NRC Emotion Lexicon [118] and SentiSense [43] can cover more words and more emotional categories with reduced cost, but may not come with the same accuracy, compared to expert annotation. For example, the NRC lexicon contains crowdsourced binary annotations of 8 emotion categories for 24,200 word senses, about 14 times more than the number of emotion words in WordNet-Affect. Both the NRC annotations and SentiSense contain binary labels instead of real numbers.

Finally, there are automatic approaches that determines sentiments or emotions of words based on their relations or associations with known words. Turney and Littman [191] measure the association between words using pointwise mutual information (PMI) and latent semantic analysis (LSA). First, an appropriate size of word neighborhood is selected, which could be a document, a paragraph, a sentence, or a fixed window of words. All words in the neighborhood are considered to be related. We can compute the PMI based on the probabilities of two words appearing in the same neighborhood. The semantic orientation, or valence of a word, is computed as its association with a given set of positive words (W_+) minus its association with a

[4] Available at http://www.unipv.it/wnop

given set of negative words (W_-).

$$\text{SO-PMI}(word) = \sum_{w_+ \in W_+} \text{PMI}(word, w_+) - \sum_{w_- \in W_-} \text{PMI}(word, w_-) \tag{1}$$

A second measure computes a singular value (SVD) decomposition of a td-idf (Term Frequency Inverse Document Frequency) matrix where rows represent words and columns represent neighborhoods. A low rank approximation of the matrix can then be obtained. The similarity between two words (LSA-Sim) is computed as the cosine of the angles between two word vectors in the U matrix of the SVD low rank approximation.

$$\text{SO-LSA}(word) = \sum_{w_+ \in W_+} \text{LSA-Sim}(word, w_+) - \sum_{w_- \in W_-} \text{LSA-Sim}(word, w_-) \tag{2}$$

Lu *et al.* [108] compute context-sensitive sentiment of words from user reviews. They point out the word "unpredictable", as an example, is usually negative in the domain of electronic appliances, but may carry positive sentiment in movie reviews. They consider multiple criteria, such as ratings on user reviews, linguistic features, and general-purpose sentiment values, and framed the problem as a linear integer programming problem. However, the NP-hardness of integer programming suggests this approach may not scale well to very large corpus.

SentiWordNet [51] is another automatic approach that propagates known sentiments of a few words to other words along inter-synset relations in WordNet. SentiWordNet 3.0 has very broad coverage, providing sentiment labels for 117,659 synsets, about 5 times as large as the NRC lexicon. Each synset is labeled with its positivity, negativity, and objectivity as a point on the 3-dimensional simplex. That is, the three dimensions must sum to 1. However, our experiments indicate SentiWordNet contains substantial amount of inaccuracy, hindering accurate detection of sentiments. To overcome this issues, Section 4.4 describes a corpus-based smoothing technique, called Smooth SentiWordNet, to alleviate the inaccuracy of SentiWordNet.

Compared to Turney and Littman, my corpus-based method takes the distance between words into account. The PMI and LSA methods consider all words in the same neighborhood to have an equal degree of relatedness. I use Gaussian kernel functions to assign less influence to words farther away. Given that sentiments may change depending on the context, I used a corpus containing only fictions, so my approach may be considered to be a narrative-specific expansion of the SentiWordNet lexicon. Mohammad [117] computes emotions associated with fictional characters using 5-grams from the Google N-Gram dataset. In comparison, I used 100-word neighborhood windows to capture longer-distance influences.

2.4 Automatically Acquiring Script-Like Knowledge

The learning of discourse structure and relationships between events has been studied for a long time. Several annotated corpora [23, 140, 144] have been created and often used in supervised approaches. As annotated corpora are difficult to create and can overfit to specific domains, in recent years there has been increasing interest in learning script-like knowledge directly with little or no supervision. In this section, I review unsupervised approaches for learning script-like knowledge.

2.4.1 Learning from General-Purpose Corpora

The earliest work on automatically extracting script-like knowledge that I am aware of is by Fujiki *et al.* [62], who worked with Japanese news corpora. They proposed three principles: (1) an action is a triplet of subject, verb, and object; (2) shared subjects and objects indicate relations between actions; (3) the significant relations can be identified by their frequency. Fujiki *et al.* used the first paragraph of news articles because they are likely to be in chronological order. Although they extracted only a small number of actions (41 pairs from 11 years of newspaper articles), the three principles are adopted by many later work.

Brody [19] proposed a similar approach for learning actions (which Brody referred to as clauses) and relations between actions. Instead of using a thesaurus like Fujiki *et al.*, Brody identified similar subjects, verbs and objects by clustering them based on co-occurrence, which can be considered as distributional similarity. Further, Brody proposed filtering valid relations between actions using the technique of hypothesis testing.

Chambers and Jurafsky [26, 27] aim at learning Schankian scripts with events and temporal partial orderings. Extending Brody's work, they identified co-occurrence of events based on pointwise mutual information and overlapping participants. Schema learned are used for the task of understanding by extracting the slot-fillers in these schema [28].

Talukdar *et al.* [180] learned clauses such as *actedIn(actor, film)* and *wonPrize(film, award)* and their relative orderings. The orderings are based on their position in the documents. As propositions in documents are not always in chronological order, they focus on macro-level propositions, which are less likely to be described in a non-chronological order.

Several work [69, 71] aimed at mining causal relations instead of entire scripts. Girju [69] used supervised learning to identify linguistic rules that indicate causal relations between noun phrases. The rules are based on WordNet categories that the noun phrases belong to and the choice of the verb. Gordon *et al.* [71] describe an approach to mining causal relations from millions of personal webblog stories. They note the challenges associated with extracting causal, commonsense information from such a corpus and also note that increasing the size of the corpus from one million to ten million produced statistically insignificant improvements. Gordon *et al.* further suggest that causal information in stories from these sources is best left implicit, and that the ability to select between causal relations does not constitute a full solution to open-domain commonsense causal reasoning.

Learning from general-purpose corpora has the advantage of large data sets, but there are also several challenges that are often encountered by learning from general-purpose corpora, which deserve some consideration. These challenges include

1. The content of general-purpose corpora may not align with the purpose of learning. These corpora may not contain the information needed, but also contain a plethora of irrelevant information. Depending on the nature of the corpora, they may be biased towards some topics and disregard other topics. For example, a corpus of news articles will contain plenty information about politics and the economy, but little about knights and goblins. Aiming at adult readers with sufficient background knowledge, news articles may not contain typical procedures for dining at a restaurant, but may contain a large number of rare incidents at restaurants. Kasch and Oates [85] attempted to tackle this challenge by retrieving documents relevant to a given topic from the Internet. Words highly correlated with the given topic are found by Latent Semantic Analysis and used to retrieve relevant documents.

2. Documents are not always written in chronological order, creating obstacles for the identification of temporal sequences. News articles usually start with a summary, and lay out the details later. They may also refer to earlier events at any time. Flashbacks and foreshadowing are common literary devices. To alleviate this problem, Fujiki *et al.* [62] only used the first paragraph of news reports. Talukdar *et al.* [180] focused on macro-level propositions.

3. Finally, the complex natural language in general-purpose corpora is difficult for current technologies to process. Although some aspects of language, such as part-of-speech, can be computationally recognized with extremely high accuracy, many important tasks such as semantic role labeling and coreference resolution remain challenging at this time.

A possible method to circumvent or alleviate these issues is to create specialized corpora with crowdsourcing. In this dissertation, the specialized corpora is created by asking crowd workers to write stories segmented into chronological sequences of events and use simple language. Such a corpus contains the information we need and relatively little irrlevant information. Learning from such specialized corpora could provide bootstrapping for learning from more complex and noisy corpora. Incremental learning has been shown to be beneficial in both cognitive science [48, 88] and deep learning research [10].

My approach adopted some statistical methods proposed by earlier work, such as the use of hypothesis testing for identifying temporal relations. This work has two specific contributions: (1) The learned knowledge is put to the test of applications of story generation and understanding, where previous work did not test the learned knowledge with extensive applications. (2) This work identifies mutual exclusion relations, which none of the above work recognizes. Mutual exclusion relations allows us to incorporate multiple possibilities in one situation, thereby creating a compact representation. For the purpose of storytelling, mutual exclusions play an important role in maintaining story coherence. Empirical evaluation of the importance of mutual exclusion relations can be found in Section 4.2.

2.4.2 Crowdsourcing Knowledge

Crowdsourcing has also been used as an effective method of knowledge acquisition. The MakeBelieve system [106] extracts commonsense rules about action sequences from the OpenMind knowledge base [168], which was built purely by crowdsourcing. However, the knowledge focuses on declarative facts about non-fictional topics, which limits their applicability to storytelling.

Regneri *et al.* [147] acquire scripts from crowdsourced bullet-list style instructions entered by anonymous individuals and then manually converted to a canonical form. They used an algorithm for aligning multiple sequences, yielding a total ordering of events. In contrast, my approach builds a partial order of events with different branches (as indicated by mutual exclusion relations). One partial order can describe multiple total orders, allowing the story generation and understanding process to dynamically adapt ordering of events depending on input or environmental changes.

Weltman *et al.* [198] designed a user interface aimed to collect highly detailed step-by-step description of pictorial stories as well as causal interpretation. However, their initial user study indicates the process to be complex and time-consuming. After two hours of training, two of the ten participants could not finish the task, and four other participants "agreed to do the bare minimum". Though detailed knowledge can support powerful NI, it is arguable that this knowledge representation errs on the side of being too difficult to acquire.

2.5 Open Narrative Intelligence

If we can supplement Narrative Intelligence systems with the ability to learn knowledge, we obtain Open Narrative Intelligence systems. These systems tackle the knowledge bottleneck that has troubled traditional approaches for decades and extend Narrative Intelligence into previously unknown domains and situations. This section reviews these systems.

As a precursor to today's Open NI systems, AQUA was proposed [145] to alleviate the knowledge bottleneck faced by story understanding systems like SAM. At a time when machine learning was still in its infancy, Ram points out that it is inherently difficult to encode sufficient knowledge for story understanding in the real world. Instead of answering questions raised by users, AQUA is capable of raising questions whose answer can improve its scripts. AQUA first tries to apply a script to understand a story. When the script contains gaps or the story contradicts the script, AQUA creates a question and stores it in memory, which may be resolved when new information comes in. AQUA can thus learn new knowledge from stories to improve the scripts it possesses. Taking a symbolic approach, AQUA is still heavily reliant on manually coded knowledge to raise appropriate questions.

SayAnything [177] is a textual case-based reasoning system that is capable of taking turns to write a story together with a user, each writing one sentence at a time, without human-authored domain knowledge except some training data. Trained on a human-annotated corpus, the system mines more than 1.6 millions stories with an average length of 26.24 sentences from Weblog entries released by spinn3r.com. After a user inputs a sentence, the n most similar sentences are retrieved using the PL2 scoring function in the Terrier Information Retrieval Toolkit [131]. A neural network was trained to rank the n sentences and the best is selected. During the training process, the entity grid feature for local coherence [9] is shown to be the best feature for training the neural network. The retrieved sentence is adapted to make well-formed and coherent sentences and make pronouns agree between sentences. The coherence of the generated story is maintained by both the sentence-retrieval mechanism and the human input.

Another textual case-based reasoning approach has been presented by Sina *et al.* [166], who modify crowdsourced semi-structured stories to create alibi for virtual suspects for the purpose of training police officers. Each crowd worker is instructed to write a three-part story, including an introduction of who, when, and where, the body of the story, and the author's overall opinion of the entire experience (e.g. of a restaurant she visited). Asking the crowd to provide semi-structured data is an effective strategy also used by my approach. The retrieval of cases depend on a manually designed feature vector. Some features, such as number of children, appear specific to the domain of application. Manually authored domain knowledge also has been used in the adaptation, such as replacing daughters with nieces.

In a work similar to this dissertation, McIntyre and Lapata [114] learned schema, or linear sequences of events that share the same entity from fairy tale texts. Each sentence is considered to be one event. Sentences with the same verb and similar arguments can be merged into one event, where similarity is computed with the Wu and Palmer's similarity measure [204] based on WordNet. The user specifies required entities in a story, and schema associated with these entities are merged into a single plot graph, where each path through the graph constitutes a story. A genetic algorithm is used to optimize for the entity grid coherence measure [9], similar to the SayAnything system.

My plot graph used in this dissertation assumes a partial order between events, and different alternatives are explicitly represented using mutual exclusion relations. McIntyre and Lapata's plot graph contain implied mutual exclusion relations between different paths in the graph. With their representation, we may generate a story containing parts of two schema only if the two sequences share an event. The increased separation is useful for learning from text corpus containing very different texts that are likely to be incompatible. In comparison, SCHEHERAZADE learns from specialized corpora containing texts that describe similar situations and are mostly compatible, allowing a greater degree of recombination.

The Restaurant Game [130] uses human playing traces in a virtual restaurant to learn a probabilistic model of restaurant activity. As the Restaurant Game is an existing virtual game, there is an *a priori* known set of actions that can occur in restaurants (e.g., sit down, order, etc.). Nevertheless, building a virtual environment manually for each possible domain in order to gather knowledge about the domain is very labor-intensive and unrealistic. One advantage of the approach developed in this dissertation is that it does not require any 3D virtual environment.

Although the above systems aim to reduce the need for manually authored knowledge, arguably none really eliminates the need completely. Manually authored and designed domain-specific knowledge appears in the form of training annotations, feature design, 3D virtual worlds, etc. Neither does the system presented in this dissertation eliminate such needs. In order to crowdsource stories pertaining to a particular social situation, a human-readable description of the situation must be written, which requires at least some knowledge about the social situation.

Lastly, it is worth mentioning some large-scale efforts for data annotation that aim to support Narrative Intelligence. Elson [50] developed an annotation scheme for stories. Analogies and similarities can be identified from annotated stories, which was shown to be a better measure of similarity compared to semantic similarities based on logic propositions. Finlayson [55] learned Propp's narrative functions from a corpora of fully annotated Russian folklore by merging story paths in a Markovian model. The system successfully identifies major narrative functions, but also misses many transitions between the events. The system has not been used to generate stories or understand unannotated stories.

To my best knowledge, this work is the first system that learns structured knowledge for any domains that can support story generation, storytelling and story understanding.

CHAPTER III

LEARNING DOMAIN MODELS

> *Listen to great storytellers; slowly, you will learn about voice, timing, tension, structure, climax—all the things you need to tell stories that will capture the imagination of your audience.*
>
> — Carmen Agra Deedy

Narrative Intelligence is highly knowledge intensive. Most existing computational NI systems rely on manually coded knowledge and are restricted to operate in a few domains where knowledge is available. The Scheherazade system is capable of learning knowledge needed for computational NI from crowdsourced exemplar stories. In this chapter I present the crowdsourcing procedures and learning algorithms.

Before any learning can take place, we collect some exemplar stories from human workers on Amazon Mechanical Turk (AMT). Collecting stories from human writers has the following advantages: First, these stories provide access to distributed memory of real-world experiences and cultural conventions as consensus among members of a society. Thus, they allow us to learn about social and cultural norms in addition to procedural knowledge. Second, it taps collective creativity that can often produce more diverse stories than an individual writer [110]. Third, we can be confident that the corpus contains highly specialized information about a specific situation. Four, we can guide the human writers to present the information in a form easy for computers to process.

This chapter provides a detailed procedure for learning domain models, or *plot graphs*, that describe how events typically unfold in social or procedural situations. I first introduce the formal representation of plot graphs in Section 3.1. After that, I describe how to acquire corpora of exemplar stories in Section 3.2. Subsequently, Section 3.3 explains how we identify primitive events in a social situation by clustering sentences with similar meaning. Section 3.6, 3.7, and 3.8 explain the learning of plot graph structures, including (1) precedence relations, (2) mutual exclusion relations and (3) optional and conditional events. The learn plot graphs are evaluated in Section 3.4 and 3.9.

3.1 The Plot Graph Representation

A plot graph describes how social and procedural situations unfold as sequences of events. In other words, a plot graph indicates what events can happen in a situation, and constrains the way in which the events can happen. In a perfect model, only event sequences that are legal according to the plot graph can happen in the situation. In learning the plot graph, we aim to accurately model real-world situations as reflected in the exemplar stories and filter out noises contained in the exemplar stories.

Definition 6 (Plot Graph)**.** *A plot graph G is a tuple $\langle E, P, M_x, E_o, E_c \rangle$. E is the set of events. $T \subseteq E \times E$ is a set of precedence relations between events. Mutual exclusions relations between the events belong to the set $M_x \subseteq E \times E$. Finally, $E_o \subset E$ is a set of optional events and $E_c \subset E$ is a set of events that are conditioned on the optional events.*

In the graphic representation, an event is represented as a vertex in the graph, a precedence relation is represented as a directed edge, and a mutual exclusion relation is a bidirectional edge. The precedence relations and the events form a directed acyclic graph (DAG).

The precedence relations and mutual exclusion relations put constraints on how a situation may unfold. A precedence relation from event $e_i \in E$ to event $e_j \in E$ indicates that event e_i always happens before event e_j in all legal event sequences according to plot graph G. A mutual exclusion relation between event e_i and event

e_j indicates that events e_i and e_j never happen in the same legal event sequences according to plot graph G.

Some graph notations and terminologies will become useful later in this chapter and are worth introducing here. We use the notation $(e_i, e_j) \in T$ to denote the fact that there is a precedence relation from event $e_i \in E$ to $e_j \in E$. We use the notation $(e_i, e_j) \in M_x$ to denote the fact that there is a bidirectional mutual exclusion relation between event $e_i \in E$ to $e_j \in E$, so $(e_i, e_j) \in M_x \Leftrightarrow (e_j, e_i) \in M_x$. We further define the following:

Definition 7 (Parent)**.** *If and only if there is a precedence relation from event $e_i \in E$ to event $e_j \in E$ (i.e. $(e_i, e_j) \in T$), event e_i is called a parent of event e_j.*

Definition 8 (Child)**.** *If and only if event e_i is a parent of event e_j, event e_j is called a child of event e_i.*

Definition 9 (Predecessor)**.** *The predecessor relation is defined as the transitive closure of the parent relation. That is, if $e_i \in E$ is a parent of event $e_j \in E$, e_i is a predecessor of event e_j. If $e_k \in E$ is a parent of event e_i and e_i is a predecessor of event e_j, e_k is also a predecessor of event e_j.*

Definition 10 (Successor)**.** *Similar to the predecessor relation, the successor relation is defined as the transitive closure of the child relation.*

Definition 11 (Path)**.** *A path in a plot graph from event e_i to event e_j is defined as a sequence of events $\langle e_1, e_2, ..., e_p \rangle$ such that every pair of adjacent events are connected by a precedence relation, i.e. $\forall q \in [1, n-1], (e_q, e_{q+1}) \in T$ and $e_1 = e_i$ and $e_p = e_j$.*

On a historic note, The partial-order DAG representation has been employed in interactive narrative systems like Weyhrauch's Tea for Three [199], and Nelson and Mateas's Anchorhead [121] to represent plots. Both systems use the directed edges to denote necessary preconditions, which means a parent must happen before its children are allowed to happen. Nelson and Mateas introduce an additional AND/OR label that covers all parents for one event. If several parents are OR-ed together, only one is needed before the child can happen. If they are AND-ed, all are required. In this dissertation, I introduce the mutual exclusion relation, which serves a similar function as the AND/OR label but is more expressive.

The main purpose of this representation is to bridge the learning process with applications of Narrative Intelligence. This representation does not model causal relations commonly used by story planners (e.g. [99]) because computationally identifying these relations remain difficult. Compared to simpler representation such as used by Chambers and Jurafsky [26], I represent mutual exclusion relations, which are important for maintaining story coherence during generation.

3.2 Collecting the Exemplar Stories

We first present the protocols for collecting exemplar stories from human workers on Amazon Mechanical Turk. Each human worker is given a short description of a social or procedure situation, such as "John and Sally went on a date at a movie theater", and are asked to write a story containing 10-20 sentences describing the entire sequence of events. A compensation from $0.60 to $1 was paid out for each accepted story.

For humans, writing stories is a natural form of communication. Telling stories is found to be an effective means for human experts to share tacit knowledge [77], which could be difficult to articulate otherwise. This is in contrast to, for example, letting workers write production rules or manipulate probabilistic graphical models and semantic networks. Turning knowledge engineering into story writing simplifies the task and helps to increase the number of potential participants and lower the cost of acquiring the corpus (i.e. the cost of hiring crowd workers to create the corpus). This allows us to easily crowdsource exemplar stories. Although the AI techniques in this dissertation are not tied to a specific story collection procedure, crowdsourcing provides a practical and cost-effective means for collection.

Understanding free-form natural language is still largely an open problem. Thus, we reduce the difficulty of natural language processing by asking the writers to use simple language. Specifically, we require the AMT crowd workers to:

- Segment the narrative into events such that each sentence contains a single activity.

- Make sure to include a complete sequence of events from the beginning to the end.

- Use proper names for all the characters in the task. This allows us to avoid co-reference resolution altogether. We provide a cast of characters for common roles, e.g., for the task of going to a fast-food restaurant, we provide named characters in the role of the restaurant patron, the waiter, and so on. Currently, these roles must be hand-specified, although future work could extract those from commonsense knowledge bases.

- Use simple natural language such as using one verb per sentence, avoiding conditionals, complex, and compound sentences. Direct speech, such as "John said 'pass me the salt' ", usually contains more than one verbs and is advised against.

- Use the past tense. Some verbs in the present tense, such as "purchase", can sometimes be misclassified as nouns by the syntactic parser. The past tense alleviates this problem.

We do not expect every human writer to follow these instructions precisely. Minor issues, such as a few misspellings or erroneous use of pronouns, are manually corrected. If the story contains too many violations of the requirements, it is rejected and no compensation is paid. A crowsourced correction is possible, as much research

Table 2: Two crowdsourced exemplar narratives in the bank robbery situation

Story 1	**Story 2**
John walked into the bank.	Sally stood behind a counter.
John went up to the counter.	John entered the bank.
John pulled out his gun.	Sally saw John.
John asked Sally for the money.	John approached Sally.
Sally started to cry.	Sally said hello.
The police arrived.	John handed Sally a note.
The police handcuffed John.	John showed Sally a gun.
The police took John away.	Sally read the note.
	Sally opened the cash register.
	Sally put money into a bag.
	Sally triggered the silent alarm.
	Sally handed John the bag.
	John thanked Sally.
	John exited the bank.

on crowdsourcing (e.g. [82, 127, 197]) have focus on this problem. However, an automated correction is out of the scope of this dissertation. At this time, lexical errors in the crowdsourced corpora are manually corrected. Due to the existence of idiosyncratic events, this correction does not completely eliminates noise in the data sets. See Section 3.3 for how the Scheherazade system handles noise in the learning the primitive events.

Experience indicates about 60-80 stories are sufficient for learning a plot graph for a situation with 40-50 events. Each story can be acquired for 60 cents to 1 dollar, so the total cost for one plot graph is around \$36 to \$80.

We refer to each segmented activity as a step. Table 2 shows two sample crowdsourced examplar stories for the situation of bank robbery. We can observe that each story mentions only some events that may happen in this situation, and they often describe the same activity using different language. In addition, the stories can describe different variations of the same situation. For example, In Story 1, John was captured before he could get the money, but in Story 2 he managed to leave the bank. These difficulties are handled by the next few steps of the learning process, starting with learning primitive events, each of which may be described using different language.

3.3 Learning Primitive Events

The event learning process discovers the primitive units of event to be included in the plot graph. This process learns salient concepts used by a community to represent and reason about common situations. This phase of learning is based on a simple assumption: salient concepts are mentioned more frequently than non-salient concepts. Therefore, if multiple sentences describe the same underlying event, the event is likely to be important for the situation. A clustering algorithm is used to seek clusters of sentences semantically similar to each other. These clusters then become events in the plot graph.

This clustering process faces two challenges. First, the algorithm must cope with small data sets with complex structures. As each crowdsourced story has a monetary cost, it is desirable to limit the number of stories that must be crowdsourced. In this dissertation, 60-80 stories are usually acquired for one social or procedural situation. Each social situation contains 30-60 event clusters. Many clusters contains only 4-5 sentences, which means the clustering algorithm must be sensitive to small clusters. Second, the algorithm must work under a substantial amount of noise. The crowd workers are not trained before the data collection, so they tend to include in their stories non-salient, idiosyncratic events, as well as non-events such as descriptions of the environments. The clustering algorithm hence must be able to filter out these sentences as noises. Please refer to Table 3 in Section 3.4 for statistics of the data sets acquired through crowdsourcing.

To cater to this highly challenging task, I adapted the OPTICS [1] clustering algorithm to extract clusters from small data sets. This method derives similarity between sentences by comparing syntactic dependency graphs, and benefits from the noise resistance of the OPTICS algorithm. This method is compared with a probabilistic method for detecting mixtures of Dirichlet distributions. Empirical results indicate that though the probabilistic method provides a principled mathematical formulation, the OPTICS-based method works better for most data sets.

In Section 3.3.1, I describe how I compute the similarity between any two sentences with their syntactic dependency graphs. In Section 3.3.2, I describe the OPTICS clustering algorithm and a new method for extracting clusters from the reachability plot it produces. In Section 3.3.3, I describe the clustering method for learning mixtures of Dirichlet distributions.

3.3.1 Sentence Similarity from Syntactic Structures

The similarity between two sentences is computed as a weighted sum of their grammatical similarity, based on their syntactic structures and similarities between the words, and temporal similarity, or how similar the two sentences are based on the relative location in the story.

I first compute the syntactic structural similarity between two sentences following Lintean and Rus [105]. The similarity between two sentences is computed as the aggregated similarities between the syntactic dependencies, which are further aggregated from similarities between words.

The Stanford parser [86] is used to extract the syntactical structure of a sentence as a directed graph (by collapsing and propagating dependencies in the basic tree structure). Each edge on the graph describes a syntactical dependency involving two words. The word at the tail of the directed edge (i.e. the word that the edge originates

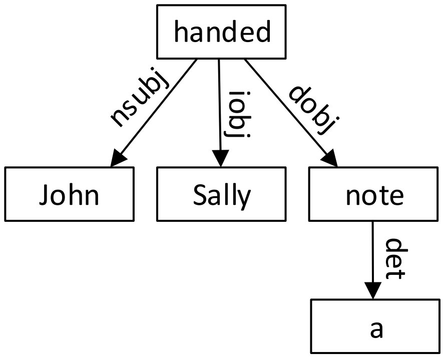


Figure 3: Parsing result of the sentence "John handed Sally a note." produced by the Stanford Parser. The grammatical relations between the words are noun subject (nsubj), direct object (dobj), indirect object (iobj) and determinant (det) respectively. The arrows point from governors to dependents.

from) is called the governor and the word at the head of the directed edge is called the dependent. A syntactic graph has one root word that has no incoming edges. Each dependency belongs to a syntactic type, such as "nsubj" or "dobj". Figure 3 shows a sample parsing result. For more information on Stanford Typed Dependencies, the interested reader is referred to [44].

When two syntactical dependencies are of different types, their similarity is zero. When the two dependencies belong to the same type, I compute the average of the word similarity between the governors and that between the dependents. The similarity between two words can be computed based on WordNet [116]. Empirically, we found the Resnik's word similarity function [148] to be the most effective.

Resnik [148] uses only the IS-A relations in WordNet. The similarity between two synsets/concepts in WordNet is computed by locating their common ancestor with the most information content, or the least probability. Between a synset s and its hypernym h, we have $P(s) \leq P(h)$. The probabilities were found from the Brown Corpus [60]. Using the information content, as argued by Resnik, is better than simply counting edges (e.g. [204]) because different edges are not of the same semantic distance. The Resnik similarity is non-negative but does not have an upper bound. I normalize the similarities to between 0 and 1 for each data set separately.

My algorithm also caters to phrases consisting of two nouns, such as "movie theaters", which frequently appear in the data sets. Noun compound phrases are indicated by a syntactic dependency of the type *nn*, or *noun compound modifier*. When such a dependency is detected, the algorithm checks if the noun compound exists as a phrase in WordNet. If it does, the noun compound is merged into one word for the purpose of similarity calculation. Treating the two words as one phrase allows the system to compute similarities at a finer granularity.

The dependencies are weighed by their distance from the root word of the sentence. Words and phrases that describe unimportant details tend to be far away from the root word. The depth of a dependency is computed as the number of edges from the dependency's dependent word to the sentence's root word. Thus, the minimum depth of a dependency is 1. An exponentially diminishing function of depth is used as the weight. Formally, the similarity between two syntactic dependency of the same type is

$$\text{SyntSim}(d_i, d_j) = w(d_i, d_j)\, s(d_i, d_j) \tag{3}$$

where the function $s(\cdot,\cdot)$ computes the average of Resnik similarities between the governors, $\text{gov}(\cdot)$, and the dependents, $\text{dep}(\cdot)$, of the two dependencies:

$$s(d_i, d_j) = \frac{\text{ResnikSim}(\text{gov}(d_i), \text{gov}(d_j)) + \text{ResnikSim}(\text{dep}(d_i), \text{dep}(d_j))}{2} \tag{4}$$

and $w(d_i, d_j)$ is the weight of this pair of dependencies:

$$w(d_i, d_j) = \exp\left(-0.2(\text{depth}(d_i) + \text{depth}(d_j) - 2)\right) \tag{5}$$

After computing the similarities between all possible pairs of syntactical dependencies from the two sentences, I find the maximum matching between the dependencies using the Hungarian algorithm [89]. Suppose sentence A and sentence B are described with a set of syntactic dependencies $D_A = \{a_1, a_2, ..., a_n\}$ and $D_B =$

$\{b_1, b_2, ..., b_m\}$. A matching $M \subset D_A \times D_B$ pairs one element in A with at most one element in B. The Hungarian algorithm finds the maximum matching $M^* = \underset{M}{argmax} \sum_{(a_i,b_j)\in M} \text{SyntSim}(a_i, b_j)$. The similarity between the two sentences is then computed as a weighted average $\sum_{(a_i,b_j)\in M^*} \text{SyntSim}(a_i, b_j) / \sum_{(a_i,b_j)\in M^*} w(a_i, b_j)$.

The second component of sentence similarity is temporal similarity. We rely on event location—a step's location as the percentage of the way through a narrative—to disambiguate syntactically similar steps that happen at different times. This is especially useful when a situation is highly linear with little variation. For example, when going to a movie theater, one will "wait in line" to buy tickets and then may "wait in line" to buy popcorn. These two activities may share many syntactical similarities, but will differ in their locations in the narrative. I use a weighted sum of the two components of the syntactic similarity and the temporal similarity, as the overall similarity between the two sentences.

$$\text{TotalSim}(d_i, d_j) = \text{SyntSim}(d_i, d_j) + \lambda \, \text{TempSim}(d_i, d_j) \tag{6}$$

The parameter λ typically ranges from 0.3 to 0.15.

3.3.2 OPTICS for Small Data Sets

OPTICS[1] [1] is one of the most popular density-based clustering algorithms with the advantage of being resistant to noise and being able to detect clusters of different shapes and densities. The algorithm maps data points onto a sequence of reachability values where dents and valleys indicate clusters. In this section, I introduce the OPTICS algorithm and propose a new heuristic for extracting clusters from reachability plots that work well on small data sets.

The underlying intuition of OPTICS is that a cluster is formed when a number of points are close to one another. The algorithm takes as input a distance matrix

[1] In the tradition of having good acronyms for clustering algorithms, OPTICS stands for Ordering Points To Identify the Clustering Structure.

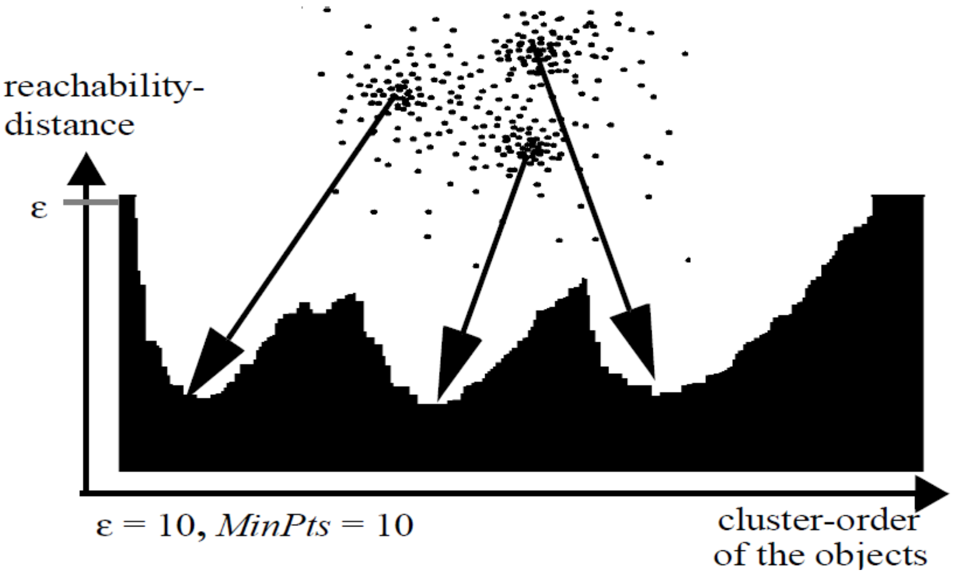


Figure 4: A reachability plot produced by OPTICS, where valleys represent clusters. Reproduced from [1]. The parameter $MinPts = C_m - 1$

that contains the distance between any two data points. It has two paramters: the minimum number of points in a cluster, denoted as C_m, and the maximum distance between two points that we will consider, denoted as ϵ. OPTICS allows ϵ to be set to infinity, which means we will consider two points however far apart they are. OPTICS computes a *core distance* for each data point, which is the radius of the circle that contains the nearby $C_m - 1$ number of points. A smaller core distance means the data point is closer to a cluster.

OPTICS defines the *reachability distance* between two points o and p as

$$\text{reachability-distance}(o, p) = \max(\text{core-distance}(p), \text{distance}(o, p)) \tag{7}$$

OPTICS can start its processing from any data point. It keeps a priority queue of all unprocessed data points, ordered from low to high reachability distance. As a new data point is processed, the reachability of all other points in the queue are updated as the reachability to the current point. OPTICS always process the data points with the smallest reachability in the queue. Therefore, when OPTICS finishes processing all data points in one cluster, which have low reachability to each other,

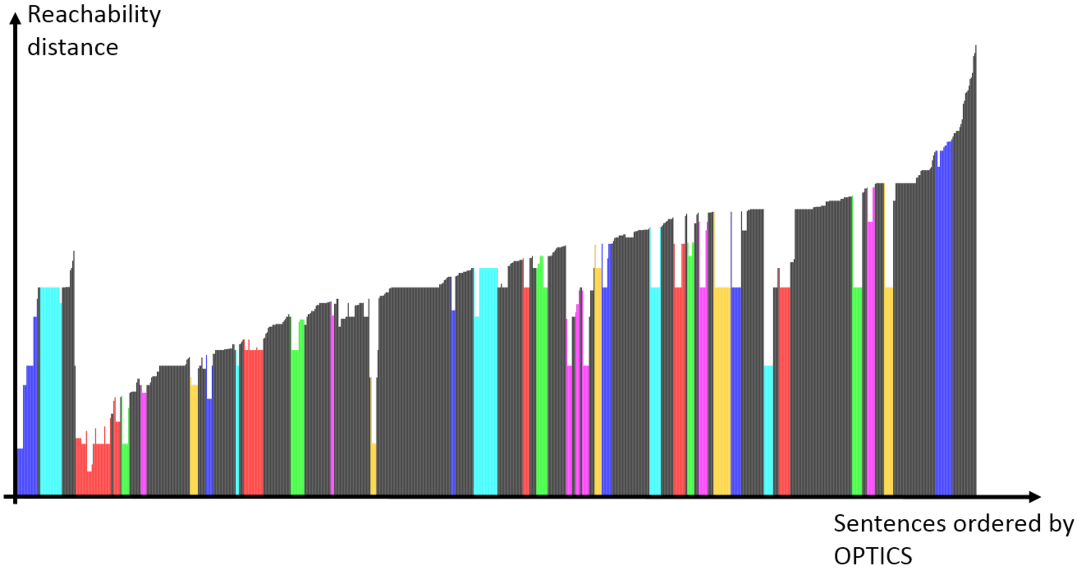


Figure 5: Clusters extracted from a reachability plot for the gas pumping situation. The recognized clusters are shown in various colors, and sentences not belonging to any clusters are shown in black.

it will encounter a data point with high reachability since this new point is far away from the current cluster. This data point could signal a new cluster or could be noise, depending on whether it is followed by other points of low reachability. When we visualize the reachability distances as a bar chart, we obtain a so-called *reachability plot*. One such reachability plot is shown as Figure 4. Regions with low reachability are clusters separated by regions of high reachability.

The next task is to extract clusters from the reachability plot. The original OPTICS paper [1] provides a method for extracting clusters from reachability plots. However, the small size of data sets we have tend to produce rather crooked reachability plots on which the technique do not work well as in more smooth plots.

In order to extract clusters from reachability plots, I first use the method by Sander *et al.* [160] to convert a reachability plot into a hierachy of clusters. I then collect all leaf nodes. Each leaf node contains a number of data points. A leaf node is considered to contain a valid cluster if (1) there is a sufficient drop in reachability between the highest point and the lowest point in the node and (2) there is a relatively flat region in the node. I then extract all regions that are relatively flat from a valid leaf node as a cluster. An example of extracted clusters is shown in Figure 5.

3.3.3 Mixtures of Dirichlet Distributions

In the second method, we represent each sentence as a bag of n-grams. I use the n-gram data produced by Lin *et al.* [103], where each n-gram is represented as a 1000-dimension vector. I normalize the original data so that each vector sums to 1. A cluster is represented as a Dirichlet distribution that all the n-grams of all the sentences in the cluster are drawn from. Formally, we parameterize each cluster with a vector $\theta_j, j \in [1, N]$. We have a number of sentences $s_i, i \in [1, M]$, each consisting of n-grams $x_1^i, \ldots, x_h^i$. Both θ_j and x_h^i are of dimension D. We maximize the overall likelihood function:

$$\Lambda(\boldsymbol{x}; \boldsymbol{\theta}) = \prod_{i=1}^{M} \sum_{j=1}^{N} p_j \prod_{h=1}^{K} P(x_h^i \mid \theta_j) \tag{8}$$

where

$$P(x_h^i \mid \theta_j) = \frac{\Gamma\left(\sum_{r=1}^{D} \theta_{j,r}\right)}{\prod_{r=1}^{D} \Gamma(\theta_{j,r})} \prod_{r=1}^{D} (x_{h,r}^i)^{\theta_{j,r}-1} \tag{9}$$

is the probability density function of the Dirichlet distribution.

This is a classic formulation where the entire data set is a mixture of distributions. The clustering problem is equivalent to finding the probabilistic model that assigns maximum probability to the observed sentences (in other words, finding the maximum likelihood estimate). This can be solved by the expectation-maximization algorithm.

To simplify the problem we introduce a set of auxiliary variables $z_{i,j} \in \{0, 1\}, i \in [1, M], j \in [1, N]$. $z_{i,j} = 1$ denotes that the sentence s_i is assigned to the j^{th} cluster. In the E-step, with θ fixed, we want to find the optimal assignment for z:

$$\begin{aligned} \hat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\operatorname{argmax}}\, P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) &= \underset{z}{\operatorname{argmax}} \prod_{i=1}^{M} \left(p_j \prod_{k=1}^{K} P(x_k^i | \theta_j) \right)^{\delta(z_{i,j}=1)} \\ &= \underset{z}{\operatorname{argmax}} \sum_{i=1}^{M} \delta(z_{i,j} = 1) \left(\log(p_j) + \sum_{k=1}^{K} \log(P(x_k^i | \theta_j)) \right) \end{aligned} \tag{10}$$

where the delta function $\delta(z_{i,j} = 1)$ equals 1 when $z_{i,j} = 1$ and equals 0 otherwise. Noting that sentences from different stories are independent, and their assignment does not affect each other given θ is fixed, we can consider the assignments of z for each story separately. Observing that two sentences in one stories rarely describe the same event, we further enforce the constraint that no sentences in one story are put into the same cluster. That is, each cluster contains one or zero sentences from each story:

$$\sum_i z_{i,j} \leq 1, \forall j \tag{11}$$

We also enforce the constraint that every sentence is assigned to exactly one cluster:

$$\sum_j z_{i,j} = 1, \forall i \tag{12}$$

This is solved as a 0-1 integer linear programming problem for each story.

In the M-step, the assignment of z is fixed, and we compute $\boldsymbol{p}$ and $\boldsymbol{\theta}$ that maximizes the data likelihood. $\boldsymbol{p}$ is computed with Laplace smoothing:

$$\hat{p}_j = \frac{\sum_{i=1}^{N} \delta(\hat{z}_{i,j} = 1) + 1}{N + M} \tag{13}$$

The the maximum likelihood estimator for $\boldsymbol{\theta}$ can be sovled for individual θ_j

$$\begin{aligned} \hat{\theta}_j = \underset{\theta_j}{\operatorname{argmax}}\, P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) &= \underset{\theta_j}{\operatorname{argmax}} \prod_{i=1}^{M} \left(\prod_{k=1}^{K} P(x_k^i | \theta_j) \right)^{\delta(\hat{z}_{i,j}=1)} \\ &= \underset{\theta_j}{\operatorname{argmax}} \sum_{i=1}^{M} \delta(\hat{z}_{i,j} = 1) \left(\sum_{k=1}^{K} \log(P(x_k^i | \theta_j)) \right) \end{aligned} \tag{14}$$

However, the maximization does not have a closed-form solution and must be solved numerically [80].

We need to select D, the number of dimensions for $\boldsymbol{x}$ and $\boldsymbol{\theta}$, so that many clusters can be sufficiently differentiated, and each θ_j can be reliably estimated from limited data. I experimentally determined $D = 150$, so that the 1000-dimension vectors provided by Lin *et al.* [103] have been reduced to 150 dimensions using principal component analysis. Each sentence was broken into the fewest number of n-grams using an A* algorithm.

In order to deal with noisy sentences, I use a large number of clusters (M typically set to about 130), and discard from the final results any clusters with less than the minimum number of sentences (i.e. C_m). This approach was shown to be effective. I also experimented with adding Markov transitions between the learned clusters to the probabilistic model. However, experiments indicate that that doing so actually decreases performance.

3.4 Evaluating the Learned Events

Corpora of exemplar stories have been collected for the following situations:

- Going to a fast food restaurant.
- Taking a date to a movie theater.
- Robbing a bank.
- Pumping gas into a car
- Making coffee
- Going through airport procedures before boarding an airplane
- Buying medicines at a pharmacy
- A wife catching her husband having an affair
- A wedding proposal

Table 3 shows the attributes of these crowdsourced corpora for each situation, including the number of stories crowdsourced, the average number of sentences in each crowdsourced story, the number of unique verbs and nouns in the entire corpus,

Table 3: Statistics of the crowdsourced corpora

Situation	Stories	Avg. Steps	Unique Verbs	Unique Noun	Gold Std. Events	Noisy Sentences	% Noisy Sentences
Fast Food Restaurant	30	6.91	41	29	21	6	2.71
Movie Date	78	11.24	108	104	56	183	20.86
Bank Robbery	60	11.78	142	115	44	160	22.63
Airport	61	15.08	130	182	49	251	27.28
Coffee Making	36	11.89	72	98	35	180	42.06
Gas Bumping	59	14.25	105	113	41	56	6.66
Pharmacy	69	12.56	146	145	58	147	16.96
Extramarital Affairs	79	14.95	248	267	59	475	40.22
Wedding Proposal	37	13.00	134	136	26	239	49.69

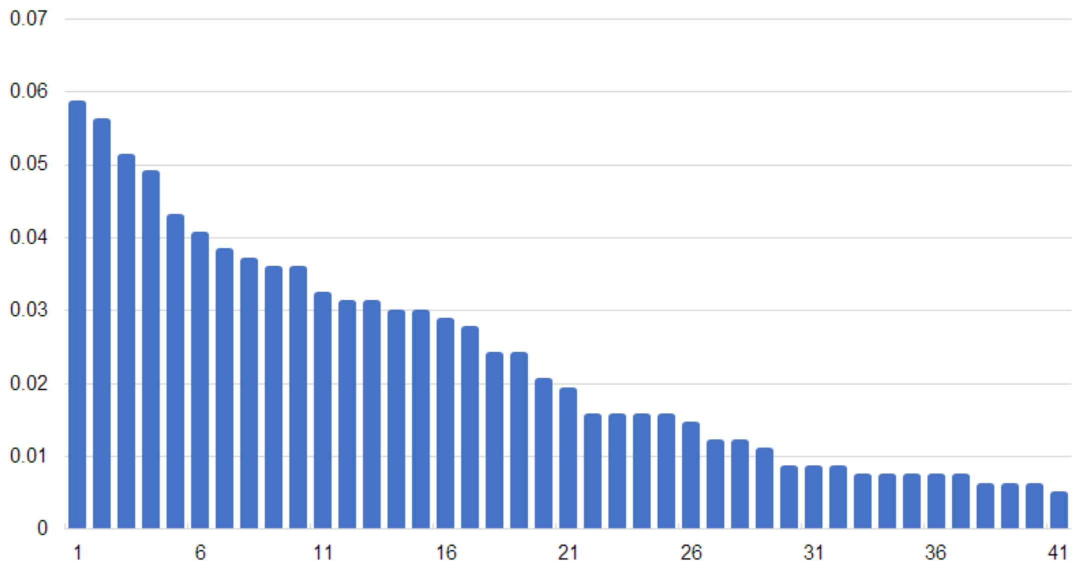


Figure 6: The size of gold standard clusters in the gas pumping situation, as percentage in the entire corpus.

and the number of events in a manually created gold standard. For each corpus, I manually created a gold standard set of clusters against which the automatically discovered clusters are evaluated against. It is worth noting that most data sets contain a large number of clusters (40-60) with only a few sentences in each cluster. Figure 6 shows the size of gold standard clusters in the gas pumping situation as their percentage in the entire data set. Only three clusters contain more than 42 sentences, or 5% of the entire corpus. On the other hand, 12 clusters contain less than 1% of sentences. Further, data sets typically contain a large percentage (about 20% to 40%) of noisy sentences that do not belong to any cluster. The compound effects of small clusters and the presence of substantial noise make the task of event learning difficult.

We compare the two learning techniques on all situations. Table 4 presents the results of event learning on our two crowdsourced corpora, using the MUC6 cluster scoring scheme [195] to match actual cluster results against the gold standard. The purity of a cluster measures intra-cluster homogeneity. Higher purity indicates higher cluster quality. For each automatically generated cluster C_i, we find a gold standard cluster G_j such that the maximum number of sentences in C_i belongs to G_j. The overall purity aggregates the overlap between C_i and G_j for all clusters. Given a set of gold standard clusters $\mathbb{G} = \{G_1, G_2, ..., G_n\}$, and a set of automatically generated cluster $\mathbb{C} = \{C_1, C_2, ..., C_m\}$, we define

$$\text{purity}(\mathbb{G}, \mathbb{C}) = \frac{1}{N} \sum_{i=1}^{m} \max_j |C_i \cap G_j| \tag{15}$$

where N is the total number of sentences.

We can observe that the best precision values for most data sets are in the range of high 70s to 80s. Two data sets, coffee making and wedding proposal, are more difficult than the rest, with precision in the 60s. The recall values are lower than precision, but it is arguable that precision is more important than recall in this task. Most purity values are in the 70s or higher, except three data sets: coffee making, gas pumping, and wedding proposal. The results indicate some data sets are more difficult than others. The two most difficult data sets, coffee making and wedding proposal, have the most noise sentences, and the second and the third least stories, which contributed to their difficulty. Performance can probably be improved by collecting more stories for the two data sets.

The first method, based on OPTICS and similarities computed from syntactic structures, consistently outperforms the second method, which models clusters as Dirichlet mixtures. This can be attributed to the several reasons: (1) the first method utilizes syntactic structures, which are more informative than n-grams, (2) the simple sentence structures resulted from crowdsourced stories helped us in computing grammatical similarities between them, and (3) OPTICS provides a robust method for dealing with noise sentences that do not belong to any clusters.

3.5 Improving Learned Events with Crowdsourcing

Although the event learning process achieves acceptably high accuracy rates, the errors at this stage can be amplified in later stages of learning, resulting in low-quality plot graphs. One possible solution is to make use of crowdsourcing again to

Table 4: Precision, recall, F1, and purity of the identified event clusters

Situation	OPTICS				Dirichlet Mixitures			
	Precision	Recall	F1	Purity	Precision	Recall	F1	Purity
Fast Food	0.880	0.688	0.772	0.836	0.753	0.738	0.745	0.642
Movie Date	0.908	0.590	0.715	0.731	0.699	0.738	0.718	0.672
Bank Robbery	0.851	0.586	0.694	0.722	0.564	0.576	0.570	0.552
Airport	0.798	0.437	0.562	0.736	0.492	0.585	0.534	0.453
Coffee Making	0.633	0.362	0.460	0.625	0.404	0.443	0.423	0.458
Gas Pumping	0.672	0.422	0.519	0.573	0.493	0.439	0.464	0.443
Pharmacy	0.778	0.512	0.617	0.705	0.532	0.526	0.528	0.484
Extramarital Affairs	0.815	0.404	0.541	0.765	0.320	0.516	0.395	0.290
Wedding Proposal	0.637	0.541	0.585	0.634	0.378	0.498	0.429	0.426

improve the clustering accuracy.

The following is one possible procedure for crowdsourcing the clustering of sentences. After the algorithm decides on a set of clusters of sentences, the clusters are shown to a number of crowd workers who are tasked to inspect the cluster members and pick out sentences that are dissimilar to other sentences in the same cluster. Under sufficient agreement, a particular sentence can be removed from its cluster. The removed sentences are put into a trash can. In the next step, we ask the crowd workers (possibly different workers) to pick out one sentence as a summary of this cluster. Finally, workers are asked to restore sentences in the trash back into the clusters wherever possible, with the help of the cluster summaries.

Preliminary results in the bank robbery domain show such a procedure may improve the purity of clusters to 89.8%, approximating agreement between humans. However, determining the best method for crowdsourcing clustering is out of the scope of this dissertation.

3.6 Learning precedence relations

Once we have discovered events that can happen in a given situation, the next stage is to identify the structure of the plot graph that most accurately explains the set of crowdsourced exemplar stories. The process begins with the identification of precedence relations, which creates a partial ordering among the events. A partial order does not require every event to be ordered with respect to every other event. Some events may be unordered with respect to each other. When a story is generated, a total order of the events consistent with the specified partial order can be selected dynamically. Hence, a partial order allows sequences of events to change according to the need of story generation or understanding and is less rigid compared to a total order. The assumption of partial order is different from approaches that learns total orders, such as Regneri *et al.* [147].

Chambers and Jurafsky [26, 27] also tried to infer temporal relations in the models them learn. They train their models on the Timebank corpus [144], which uses temporal signal words. Since our corpora are highly specialized to our task, the precedence relations can be inferred directly from the sentence orders in the exemplar stories.

I present two methods for learning the precedence relations. The first method applies a global threshold to the confidence of all possible precedence relations, and then attempts to restore precedence relations that might have been left out by a conservative threshold. The second method keeps as many precedence relations as possible, while maintaining the acyclicity of the graph. In practice, I find both methods have their merits. The first method helps to show the major relations in the graph, allowing us to identify important graph structures. Graphs learned by the first method can produce a greater variety of stories. The second method eliminates cycles while preserving more precedence constraints. It produces more linear stories with less variety, which helps to reduce errors in generated stories.

The precedence learning process selects valid precedence relations among all possible relations. I present two different methods for learning precedence relations, and both relies on the notion of the confidence of each relations, as defined below.

For every pair of events (e_i, e_j), there are three possible ordering: e_i is ordered before e_j; e_i is ordered after e_j; or the order between e_i and e_j is not important for the situation. The three cases are denoted using the following mathematical notations: $e_i \prec e_j$, $e_i \succ e_j$, and $e_i \parallel e_j$. The reader may have noticed that the notation $e_i \succ e_j$ is equivalent to $e_j \prec e_i$. When e_i is ordered before e_j, there may still be other events ordered between e_i and e_j. We can choose to accept or reject the two hypotheses $e_i \prec e_j$ and $e_i \succ e_j$ — only one of the two can be accepted. If both are rejected, we accept $e_i \parallel e_j$.

We count the amount of evidence for and against the two hypotheses. Let s_l be a sentence in the event cluster e_i, and s_m a sentence in the event cluster e_j. If s_l and s_m both appear in the same exemplar story, and s_l is ordered before s_m, we count one observation in support of $e_i \prec e_j$. Conversely, if s_l and s_m appear in the same exemplar story and s_m appears before s_l, we count one observation in support of $e_i \succ e_j$.

The confidence of $e_i \prec e_j$ is computed by a one-tailed hypothesis testing based on the binomial distribution as

$$\text{confidence}(i, j) = \sum_{i=1}^{k_{ij}-1} \binom{n_{ij}}{i} \frac{1}{2^{n_{ij}}} \tag{16}$$

where n_{ij} is the total number of observations we have, and k_{ij} is the observations that support $e_i \prec e_j$. Similarly, the confidence of $e_i \succ e_j$ is computed based on the number of observations that support $e_i \succ e_j$.

3.6.1 Method 1: Smart Thresholding

The first method accepts all hypotheses $e_i \prec e_j$ whose confidence exceeds a threshold $T_p \in [0.5, 1)$. The threshold T_p applies to the entire graph and allow us to generate an initial estimate of the graph structure through simple counts. In practice, we find that the graph quality is sensitive to the selection of these parameters. Selecting a set of parameters that always work for the entire graph is often impossible as different parts of the graph may respond well to different parameters. Thus, it is desirable for graph estimation to be robust against parameter selection, and to locally relax the global thresholds for some relations. We achieve this goal by using a high threshold for T_p, which filters out many potential precedences, and then restoring missing relations back to minimize a measure of graph error, effectively relaxing the global threshold locally.

Since a plot graph encodes event ordering, we introduce an error measure based on the expected number of interstitial events between any pair of events. The error is the difference between two distance measures, $\mathrm{D_G}(e_i, e_j)$ and $\mathrm{D_N}(e_i, e_j)$. $\mathrm{D_G}(e_i, e_j)$

is the number of events on the shortest path from e_i to e_j on the graph (e_i excluded). In contrast, $\mathrm{D_N}(e_i, e_j)$ is the normative distance from e_i to e_j averaged over the entire corpus of exemplar stories.

Formally, we find the shortest path from e_i to e_j and denote it as $sp_{i,j}$. Recall a path is a sequence of events $\langle e_i, e_1, e_2 ..., e_j \rangle$ starting from event e_i, ending in event e_j, and every adjacent pair is connected by a precedence relation. Similarly, we find the shortest path from e_j to e_i and denote it as sp_{ji}. If e_i is ordered with respect to e_j, one of sp_{ij} and sp_{ji} exists. If e_i is parallel to e_j, neither sp_{ij} nor sp_{ji} exists. The graph distance between e_i and e_j is thus defined as

$$\mathrm{D_G}(e_i, e_j) = \begin{cases} |sp_{ij}|{-}1, & \text{if } \exists sp_{ij} \\ -|sp_{ji}|{+}1, & \text{if } \exists sp_{ji} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

Since the shortest path contains at least two events e_i and e_j, $\mathrm{D_G}(e_i, e_j) \geq 1$ if there is at least one path from e_i to e_j. $\mathrm{D_G}(e_i, e_j) = 0$ if e_i and e_j are unordered with respect to each other on the graph. $\mathrm{D_G}(e_i, e_j) \leq 1$ if there is at least one path from e_j to e_i.

Having defined the graphical distance $\mathrm{D_G}(e_i, e_j)$, we formally define the normative distance $\mathrm{D_N}(e_i, e_j)$ between two events e_i and e_j, as the average distance between the two events over all exemplar stories. An exemplar story N_k is a sequence of sentences $\langle s_1^k, s_2^k, ... \rangle$. Let the notation $s_1^k \in e_i$ denote that sentence s_1^k belongs to the event cluster e_i. The definition of $\mathrm{D_N}$ relies on the definition of $\mathrm{d_N}$, simply the number of interstitial sentences between any two sentences in the same narrative. That is, for any $s_a^k, s_b^k \in N_k$,

$$\mathrm{d_N}(s_a^k, s_b^k) = b - a \tag{18}$$

Next, we define the set S_{ij} as the set of pairs of sentences that appear in the same exemplar story and belong to e_i and e_j respectively:

$$S_{ij} = \{(s_a^k, s_b^k) | s_a^k \in e_i \wedge s_b^{k'} \in e_j\} \tag{19}$$

Now $\mathrm{D_N}(e_i, e_j)$ is just the average of $\mathrm{d_N}(\cdot)$ over the set S_{ij}:

$$\mathrm{D_N}(e_i, e_j) = \frac{1}{|S_{ij}|} \sum_{(s_a^k, s_b^k) \in S_{ij}} \mathrm{d_N}(s_a^k, s_b^k) \tag{20}$$

As the distance on the plot graph should approximate the average distance between the corresponding sentences in the exemplar stories, we aim to minimize the mean squared graph error (MSGE), which is defined as the average squared distance between the graph distance $\mathrm{D_G}$ from the ground truth $\mathrm{D_N}$:

$$\mathrm{MSGE} = \frac{1}{|P|} \sum_{e_i, e_j \in P} (\mathrm{D_G}(e_i, e_j) - \mathrm{D_N}(e_i, e_j))^2 \tag{21}$$

where P is the set of all ordered event pairs (e_i, e_j) such that e_j is reachable from e_i or e_i and e_j are parallel on the graph (i.e. $\mathrm{D_G}(e_i, e_j) \geq 0$):

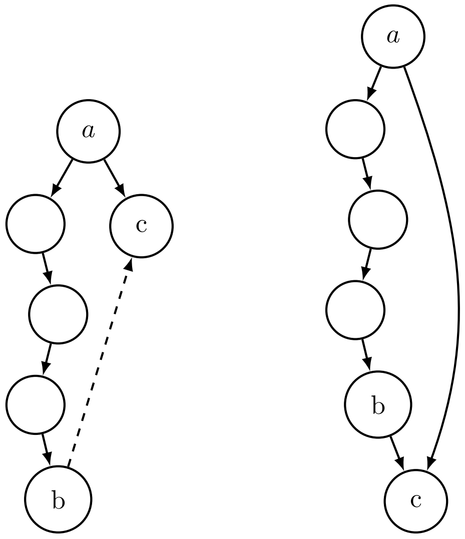
$$P = \{(e_i, e_j) | \mathrm{D_G}(e_i, e_j) \geq 0\} \tag{22}$$

We note that for some data sets, removing any outlier sentences that do not belong to any clusters from the stories before counting the interstitial sentences tends to produce better results.

We utilize the MSGE error measure to improve the graph based on the belief that $\mathrm{D_N}$ represents the normative distance we expect between events in any narrative accepted by the plot graph. That is, typical event sequences in the space of narratives described by the plot graph should have $\mathrm{D_G}(e_i, e_j) \approx \mathrm{D_N}(e_i, e_j)$ for all events. A particularly large deviation from the norm may indicate that some edges with low confidence could be included in the graph to make it closer to user inputs and reduce the overall error.

We implement a greedy, iterative improvement procedure that reduces mean square graph error in a plot graph (Algorithm 1). For each pair of events (e_i, e_j) such that e_j is reachable from e_i or the two events e_i and e_j are parallel, we compute a set of potential predecessor events U:

$$U = \{e_k | e_k \in \text{successors}(e_i), \mathrm{D_G}(e_i, e_k) = \mathrm{D_N}(e_i, e_j) - 1\}$$



(a) Before (b) After

Figure 7: Restoring low-confidence precedence relations: (a) The precedence relation (shown as dashed arrow) from b to c has a confidence below the global threshold and is initially rejected. (b) Adding the precedence back to the graph creates the desired separation between a and c.

where successors(e_i) is the set of all direct and indirect successors of e_i. In other words, if any $e_k \in U$ becomes the immediate predecessor of e_j, $\mathrm{D_G}(e_i, e_j) = \mathrm{D_G}(e_i, e_k) + 1$ will be equal to $\mathrm{D_N}(e_i, e_j)$. Starting from the pairs of events (e_i, e_j) with the largest deviation from the norm, computed as $\mathrm{D_N}(e_i, e_j) - \mathrm{D_G}(e_i, e_j)$, we check if adding an edge $e_i \to e_j$ will create any cycles or increases the graph error MSGE. If not, the edge $e_i \to e_j$ is added to the graph. This intuition is illustrated in Figure 7 where the edge (dashed arrow) from event b to event c was originally rejected due to insufficient confidence; adding the edge to the graph creates the desired separation between events a and c.

Adding an edge may increase overall graph error. However, in some domains, adding all edges that do not create cycles, regardless if adding them increases MSGE, tends to produce graphs correspond better to human intuition. This is due to factors not captured by the MSGE heuristic. If we skip that MSGE test, we obtain an aggressive version of graph improvement.

Algorithm 1 Plot graph improvement

```
procedure ImproveGraph($G = \langle E, P, M_x, E_o, E_c \rangle$)
    $P \leftarrow$ all event pairs $(e_i, e_j)$ such that $e_j \in \mathit{successor}(e_i)$ or that $e_i$ and $e_j$ are
unordered, i.e. $\mathrm{D_G}(e_i, e_j) \geq 0$.
    $P \leftarrow$ Sort $P$ in decreasing order of $\mathrm{D_N}(e_i, e_j) - \mathrm{D_G}(e_i, e_j)$.
    for each $(e_i, e_j)$ in $P$ do
        $U \leftarrow$ successors $e_k$ of $e_i$ such that $\mathrm{D_G}(e_i, e_k) = \mathrm{D_N}(e_i, e_j) - 1$
        for each $e_k$ in $U$ do
            if edge $\langle e_i, e_k \rangle \notin T$ then                          ▷ add new edge
                Build a new graph $G' = \langle E, T \cup \langle e_i, e_k \rangle, M_x, E_o, E_c \rangle$
                if G′ does not contain cycles, and MSGE($G'$) < MSGE($G$) then
                    $G \leftarrow G'$                   ▷ Aggressive version skips the MSGE test
                end if
            end if
        end for
    end for
end procedure
```

We find a relatively high T_p (≈ 0.7) combined with the graph improvement step leads to robust graph estimation. A conservative T_p initially discards many edges in favor of a lower-diameter graph with many unordered events. After that, the improvement algorithm opportunistically restores the relations of different levels of evidence as long as graph error can be reduced. This effectively relaxes the threshold locally. Rare events are automatically excluded from the graphs because their relations to all other events do not meet our probability and confidence thresholds.

3.6.2 Method 2: Integer Quadratically Constrained Programming

Even with the graph improvement step, which relaxes the global threshold somewhat, sometimes it can be difficult to find a good value for the parameter T_p. This issue is further complicated by the requirement that the plot graph must be acyclic. To solve these two issues, I propose the second method for learning precedence relations using integer programming.

To avoid looping infinitely back to the same event, any plot graph must be a directed acyclic graph (DAG). By setting $T_p > 0.5$, we can make sure to reject one of

$e_i \prec e_j$ and $e_j \prec e_i$, hence eliminating cycles of two events. There are no self cycles because no event can precede itself. However, the graph may still contain cycles that involve three or more events.

Cycles can always be eliminated by increasing the global threshold. Cheng *et al.* [32] discussed two simple methods to find the minimum threshold to eliminate all cycles from the graph. Nevertheless, increasing the global threshold affects the entire graph and may discard good precedence relations that are not involved in any cycles. Thus, it is difficult to learn an acyclic graph by only adjusting the global threshold.

In order to eliminate cycles in the plot graph and preserve as many precedence relations as possible, I formulate the learning problem as an integer quadratically constrained program (IQCP). IQCP is an NP-hard problem, but very efficient off-the-shelve solvers, such as the Gurobi solver [75], are available. The general case of cycle elimination (i.e. minimum feedback edge set) is also NP-hard and APX-hard (i.e. difficult to approximate) [84]. Thus, formulating precedence learning as IQCP is appropriate.

The formulation of the IQCP problem is based on the insight that vertices in a DAG can be arranged into a number of layers where all directed edges go from a higher layer to a lower layer. An illustrative example is shown in Figure 8, which shows a directed graph containing 9 vertices organized into 5 layers. All edges go from a higher layer (denoted with a smaller number on the vertices) to a lower layer (denoted with a greater number on the vertices) except the red edge at the bottom right, which creates a cycle. Eliminating edges going from lower layers to higher layers guarantees that the graph is acyclic.

Consequently, we can formulate the IQCP problem as follows. We attach a level variable l_i, which can take any positive integer value, to each vertice e_i. We also attach a binary variable $x_{ij} \in \{0, 1\}$ for any precedence hypothesis $e_i \prec e_j$. $x_{ij} = 1$ if and only if we accept the hypothesis. We want to preserve as many precedence

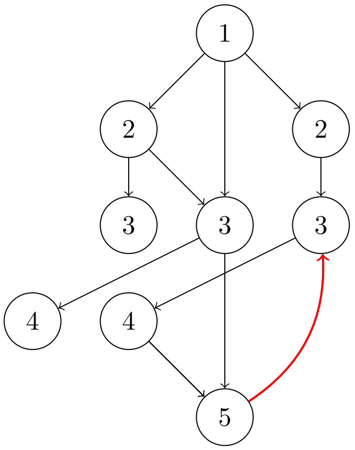


Figure 8: Organizing vertices in a directed graph into multiple layers. All edges go from a higher layer (denoted by a smaller number) to a lower layer (denoted with a greater number) except the cycle-creating red edge at the bottom right.

relations as possible and prioritize hypothesis of higher confidence, so we maximize the following sum of logarithm.

$$\boldsymbol{x} = \operatorname*{argmax}_{\boldsymbol{x}} \sum_i \sum_j \left(\log(conf(i,j)) - \log(T_p)\right)\; x_{ij} \tag{23}$$

The reason that we subtract $\log(T_p)$ is the following: $\log(x)$ is negative when $0 < x < 1$, so a direct sum of logs will be 0 or negative. However, we know T_p is a lower bound on the confidences of hypotheses that we accept. By subtracting $\log(T_p)$, we ensure $\log(conf(i,j)) - \log(T_p) > 0$, for all hypotheses we may accept. It is worth noting that adding a constant C to $\log(x)$ shifts the function upwards and does not change the shape of the function. We typically set T_p to 0.5, so any precedences with 50% or lower confidence are rejected, but a higher global threshold may also be used.

In order to ensure the plot graph is acyclic, we need to respect the constraint:

$$l_j - x_{ij}\, l_i \geq 1, \forall i, \forall j \tag{24}$$

In plain English, if we accept the hypothesis $e_i \prec e_j$, $x_{ij} = 1$, then the level variable l_j must be greater than the level variable l_i, so that the directed edge goes from a lower level to a higher level. If we reject the hypothesis $e_i \prec e_j$, the variable x_{ij} will be set to 0, and the constraint is always satisfied since l_j is a positive integer. It is clear that the objective function is linear but the constraints are quadratic with respect to the variables $\boldsymbol{x}$ and $\boldsymbol{l}$.

3.6.3 Practical Concerns In Learning Precedence Relations

It is worthwhile to compare the two methods for learning plot graphs. The smart thresholding method first filters all precedence hypotheses with a hard global threshold and then tries to compensate for it. The IQCP method typically uses a lower hard threshold and removes precedence relations relatively lower confidence whose inclusion will creates cycles. Therefore, the IQCP method takes a "softer" and more flexible approach which retains more precedence relations. This is advantageous when data are sparse. However, when data are sufficient, the IQCP method may appear too lenient and keeping incorrect precedence relations that are only slightly above the lower threshold. This may be compensated by a higher hard threshold.

In general, we face the classic trade-off between overfitting and underfitting, or bias and variance, when choosing the correct filtering threshold. A low global threshold or an aggressive precedence restoration algorithm, can accept more precedence relations, and may potentially leads to overfitting. On the other hand, if a high threshold is adopted, we may learn too few precedence relations and the model can underfit. An overfit model possessing many precedence constraints will produce very limited variations in story generation, whereas an underfit model will produce variations that are too wild to make sense. As with most machine learning algorithms, it is generally difficult to presciently determine the correct trade-off that works on all data sets and all situations.

Since IQCP is NP-hard, it may not scale well to very large problems. However, it works well for the size of the problems I deal with. As I showed in Table 3, all situations have less than 60 gold standard events. Counting one level variable l for each event and one acceptance variable x for each potential edge, there are at most 3660 variables, which are well within the capability of modern solvers and hardware. On an Intel i5 (Haswell architecture) processor with 16 GB RAM, the Gurobi solver [75] takes less than 1 second to solve the IQCP problems for every situation I have.

3.7 Learning Mutual Exclusion Relations

After identifying precedence relations, we further identify mutual exclusion relations between the events. Mutual exclusion relations are a generalization of the AND/OR labels introduced by Nelson and Mateas [121]. Many social and procedural situations contain several possible outcomes or alternative ways to perform an activity. For example, a bank robber may escape or be caught by police. A restaurant patron may choose to dine-in or order to-go meals. Due to the presence of alternative or branching event sequences, different exemplar stories may describe different events, and the plot graph may contain several paths that are incompatible. In order to separate incompatible events and create coherent legal event sequences, the learning process identifies mutual exclusion relations.

In the plot graph representation, a bidirectional mutual exclusion relation between two events indicates that the two events cannot both happen in a single story. Mutual exclusion relations pose constraints on valid sequences of events, which help to maintain story coherence during story generation.

Mutual exclusion relations are identified based on the mutual information between events, a measure of their interdependence. Suppose $E_i \in \{0, 1\}$ is a random variable indicating if event e_i exists in an input narrative. The mutual information between two events e_i and e_j is:

$$\mathrm{MI}(E_i, E_j) = \sum_{E_i \in \{0,1\}} \sum_{E_j \in \{0,1\}} p(E_i, E_j) \log \frac{p(E_i, E_j)}{p(E_i) p(E_j)} \tag{25}$$

where $p(\cdot)$ denotes the probability of the random variables. For example, $p(E_i = 1)$ is the probability that event e_i happens in a narrative, estimated as the ratio of input narratives containing e_i to the total number of narratives in the corpus. $p(E_i = 1, E_j = 1)$ is the probability that e_i and e_j happen in the same narrative, etc. We can also write the mutual information as:

$$\mathrm{MI}(E_i, E_j) = C(0,0) + C(0,1) + C(1,0) + C(1,1) \tag{26}$$

where

$$C(a,b) = p(E_i = a, E_j = b) \log \frac{p(E_i = a, E_j = b)}{p(E_i = a)p(E_i = b)} \tag{27}$$

The partial sum $C(0,1)+C(1,0)$ expresses the tendency for the two random variables to take on different values, or the tendency that the presence of one event predicts the absence of the other event. We recognize two events to be mutually exclusive when $C(1,0)+C(0,1) > 0$, which suggests mutual exclusion, and $\mathrm{MI}(E_i, E_j)$ is greater than a threshold T_m, which suggests strong interdependence between the two variables.

3.8 Learning Optional and Conditional Events

To explain the underlying rationale for having optional and conditional events, we must jump forward a little in content to explain one of the several rules in story generation (the topic is addressed in Chapter 4). The one rule that is relevant here is that we assume that precedence constraints are equivalent to necessary conditions. That is, if event a temporally precedes event b, event b can only happen after event a has happened because event a establishes a necessary condition for event b.

Optional events are introduced to make sure every event in the plot graph has a chance of appearing in at least one legal event sequence that may be generated from the graph. Given two events a and b, there are several possible configurations:

1. Events a and b are temporally ordered but not mutually exclusive.

2. Events a and b are not temporally ordered but mutually exclusive.

3. Events a and b are both temporally ordered and mutually exclusive.

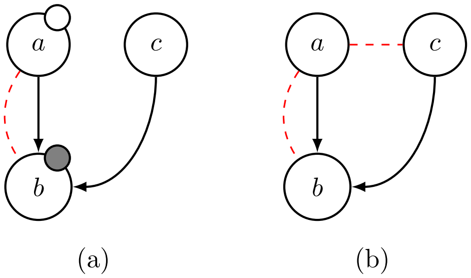


Figure 9: Identifying optional events: (a) Event a is optional and b is conditional. (b) Event a is not optional due to the mutual exclusion relation between a and b.

The first two cases can be easily handled, but the third case creates a paradox. Without loss of generality, let us assume event a is ordered before event b, or $(a, b) \in E$. Due to this precedence relation, event b cannot happen until event a has happened, but after event a has happened, event b will be excluded by the mutual exclusion relation and still cannot happen.

To make sure event b can happen in some legal event sequences, we must relax our assumption of necessity Instead of requiring event a to be a precondition for event b, we do not consider event a to be a precondition for any event. That is, event a becomes optional in the plot graph. If event a occurs in a narrative, event b will be excluded, but if event a does not occur, then event b can happen and is still necessary for its successor events. Event b becomes conditioned event a.

We need to be careful not to make events unnecessarily optional. One interesting scenario is illustrated in Figure 9. On the left, we recognize event a to be optional and event b to be conditioned on event a. On the right, however, event a is mutually exclusive to another event c that is mutually exclusive to event a and is a predecessor for event b. When event c happens, event a will be excluded from the plot graph, and event b may be included in an event sequence. Thus, in this scenario, we do not recognize event a to be optional or event b to be conditional.

Now let me introduce the notion of a *clear path*. Recall that a path from event e_i to event e_j in a plot graph is a sequence of events $\langle e_i, e_1, e_2, \ldots, e_j \rangle$, where adjacent

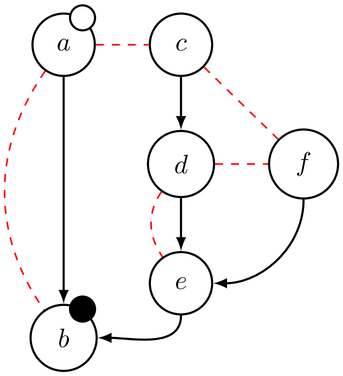


Figure 10: Identifying optional events with clear paths. Although event a is mutually exclusive to event c, a predecessor of event b, there is no clear path from event c to event b. Therefore, we must recognize event a as optional and event b as conditional.

events are connected by a precedence relation. The path is clear if no two events on the path are mutually exclusive to each other. Moreover, the path is also clear if two events e_p and e_q $(p < q)$ on the path are mutually exclusive to each other, but e_p has been recognized as optional.

In general, when we have two events a and b, in order to recognize event a as optional and event b as conditional, all of the following conditions must be satisfied:

- Event a precedes event b, or $(a, b) \in T$
- Event a is mutually exclusive to event b, or $(a, b) \in M_x$
- There does not exist event c such that event c is mutually exclusive to event a, and there is a clear path going from event c to event b, and a is not included in this path. Or formally,

$$\nexists\, c \in E \;\; s.t. \;\; (a, c) \in M_x \wedge \exists ClearPath(c, b) \wedge a \notin ClearPath(c, b) \qquad (28)$$

It should be noted that this set of conditions are sufficient, but not necessary, to make sure every event can happen in at least some event sequences.

To see why we need the notion of a clear path, let us consider the scenario in Figure 10. We notice that event f is not optional because event f provides an alternative path to event e. We also notice that event a is mutually exclusive to event c, a predecessor of event b. However, the path from event c to event b is not clear because it goes through the pair of events d and e. When event c happens, event e cannot happen, so there is effectively no path from event c to event b. Thus, we must recognize event a as optional and event b as conditional.

A polynomial-time algorithm for detecting optional and conditional events can be built using the adjacency matrix representation of graphs. The adjacency matrix M is constructed as follows:

$$M_{ij} = \begin{cases} 1 & \text{if there is precedence relation from vertex } i \text{ to vertex } j \\ 0 & \text{if there is no precedence relations from vertex } i \text{ to vertex } j \end{cases} \tag{29}$$

Let $M^2 = M \times M$, and $M^3 = M \times M \times M$, and so on. It is easy to show that if and only if there is a path from vertex i to vertex j that contains one intermediate vertex between i and j, $M^2_{ij} > 0$. Otherwise, $M^2_{ij} = 0$. This is because if there is another vertex k such that $M_{ik} = 1$ and $M_{kj} = 1$, then $M^2_{ij} = \sum_{k'} M_{ik'} M_{k'j} > 0$. More generally, we can show $M^n_{ij} > 0$ if and only if there is a path from vertex i to vertex j that contains $n-1$ intermediate vertices.

The fact that the plot graph is acyclic helps us reduce the computation needed because the length of longest path in the graph cannot exceed the number of vertices, n. Therefore, to find if there is a path between vertex i and vertex j, it suffices to check the entry at row i and column j in the matrices $M^2, M^3, \ldots, M^{n-1}$. A path exists if and only if the entry in any of these matrices is non-zero. We do not need to compute matrices to the n^{th} or higher powers, because a path containing more than $n-2$ intermediate vertices cannot exist in the acyclic graph. To make sure we find clear paths only, for each mutual exclusion relation between vertices $e_{i'}$ and $e_{j'}$, we set the entries at (i', j') and (j', i') to zero after each matrix multiplication.

The algorithm for identifying optional and conditional events is shown as Algorithm 2. The most time consuming step in the algorithm is the computation of

Algorithm 2 Identifying optional and conditional events

```
procedure IdentifyOptional($G = \langle E, P, M_x, E_o, E_c \rangle$)
    $M^1 \leftarrow$ an $n$-by-$n$ zero matrix
    for each $(e_i, e_j) \in T$ do
        $M^1_{ij} \leftarrow 1$
    end for
    for $k \leftarrow 2$ to $n - 1$ do
        $M^k \leftarrow M \times M^{k-1}$
    end for
    $P \leftarrow$ a empty list of pairs of events
    for each $(e_i, e_j) \in M_x$ do
        for $k \leftarrow 1$ to $n - 1$ do
            if $M^k_{ij} > 0$ then
                $P \leftarrow P$ append (i, j)
                break
            else if $M^k_{ji} > 0$ then        ▷ acyclicity: $M^k_{ij}$ and $M^k_{ji}$ cannot both $> 0$
                $P \leftarrow P$ append (i, j)
                break
            end if
        end for
    end for
    for each $(i, j) \in P$ do
        if $\nexists\, e_q \in E$, s.t. $(e_q, e_i) \in M_x$ and $q \neq j$ and ClearPath($G, e_i, e_q, e_j$) then
            $O \leftarrow O \cup e_i$
            $C \leftarrow C \cup e_j$
        end if
    end for
end procedure
function ClearPath($G = \langle E, P, M_x, E_o, E_c \rangle, e_i, e_q, e_j$)
    $M^1 \leftarrow$ an $n$-by-$n$ zero matrix
    for each $(e_a, e_b) \in T, a \neq i, b \neq i$ do        ▷ the path cannot go through $e_i$
        if $(e_a, e_b) \notin M_x$ then
            $M^1_{ab} \leftarrow 1$
        end if
    end for
    for $k \leftarrow 2$ to $n - 1$ do
        $M^k \leftarrow M \times M^{k-1}$
        for each $(e_a, e_b) \in M_x$ do
            $M^k_{ab} \leftarrow 0$
        end for
    end for
    return $\exists k$, s.t. $M^k_{qj} > 0$
end function
```

$n-2$ matrices $M^2, M^3, \ldots, M^{n-1}$. The simplest matrix multiplication algorithm has time complexity of $O(n^3)$. The widely used Strassen algorithm [175] has reduced the complexity to $O(n^{2.807355})$, even though more complex and faster methods have been invented (e.g. the Coppersmith-Winograd algorithm [36]). Since there are $O(n^2)$ pairs of vertices in the graph, we need to run the algorithm no more than $O(n^2)$ times. Hence, it only takes polynomial time to find all optional and conditional events.

3.9 Evaluating the Learned Graphs

Before applying learned plot graphs in tasks of Narrative Intelligence, it is desirable to check if the graphs make sense to human judges. In this section, I perform a quantitative evaluation on some of the learned plot graphs. The learned precedence relations in the plot graphs are evaluated by crowd workers from AMT. The mutual exclusion relations and optional events are not quantitatively evaluated, as their meaning can become opaque when evaluated without a proper context. However, they will be indirectly evaluated when we evaluate the generated stories in Section 4.2.

Figure 11 and Figure 12 show the plot graphs learned by the smart thresholding and the IQCP method respectively (referred to as the ST graph and the IQCP graph hereafter). These plot graphs are learned from the gold standard clusters under the assumption that we can further crowdsource the clustering of sentences and achieve near perfect clusters. The event labels are English interpretations of each event for presentation purposes only, based on manual inspection of the sentences in each event. For clarity, edges that do not affect the partial ordering are omitted from the figure.

Close inspections of the plot graphs show that the two graphs do not differ in the sequence of major events. The ordering of the key events, such as “buy tickets”, “find seats”, “watch movie”, and “go home”, are mostly correct in both graphs. Most precedence relations seem to be reasonable, although a few can be added or removed to make the graphs to capture the ideal “date at a movie theater” situation

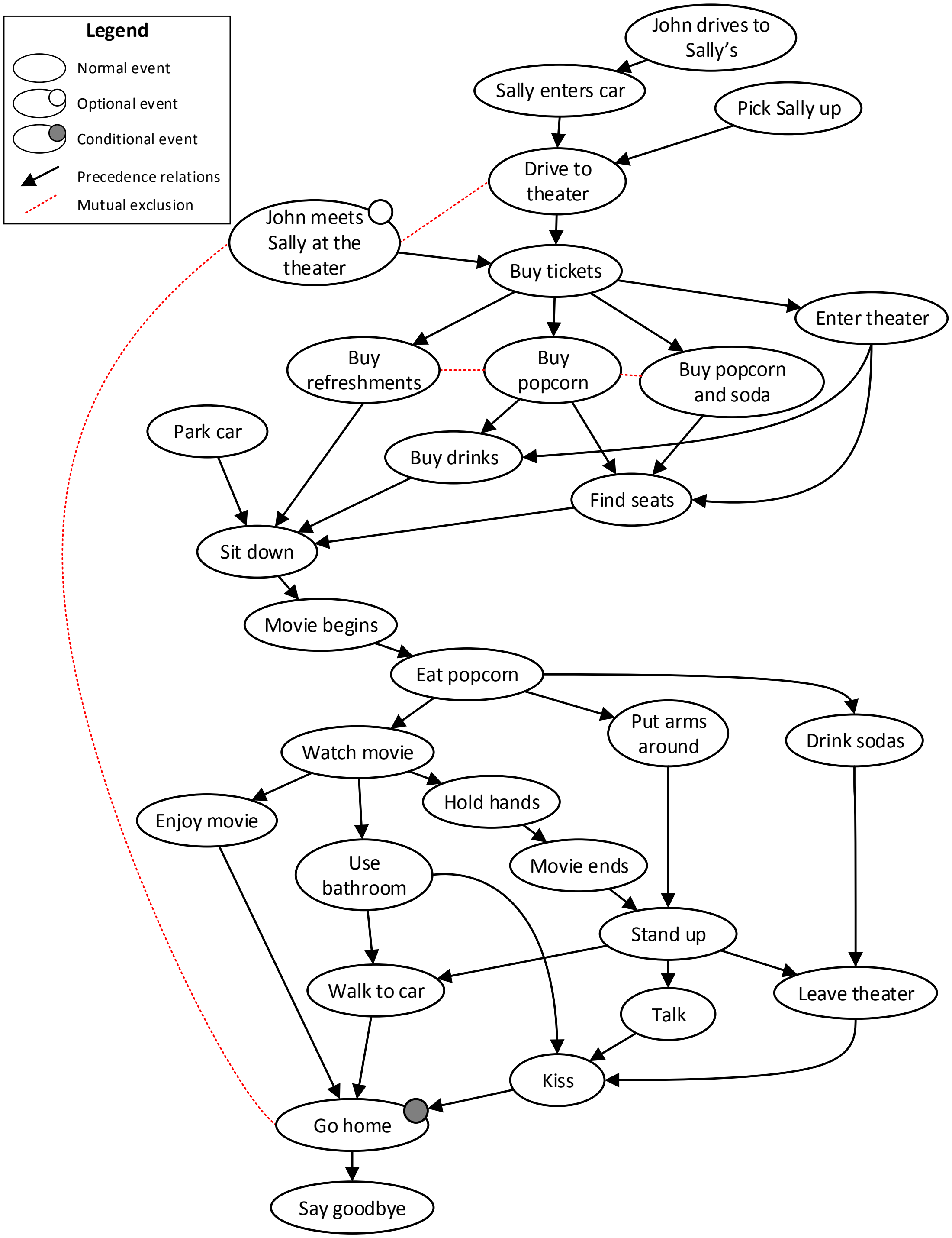


Figure 11: A plot graph for the movie date situation, created by smart thresholding.

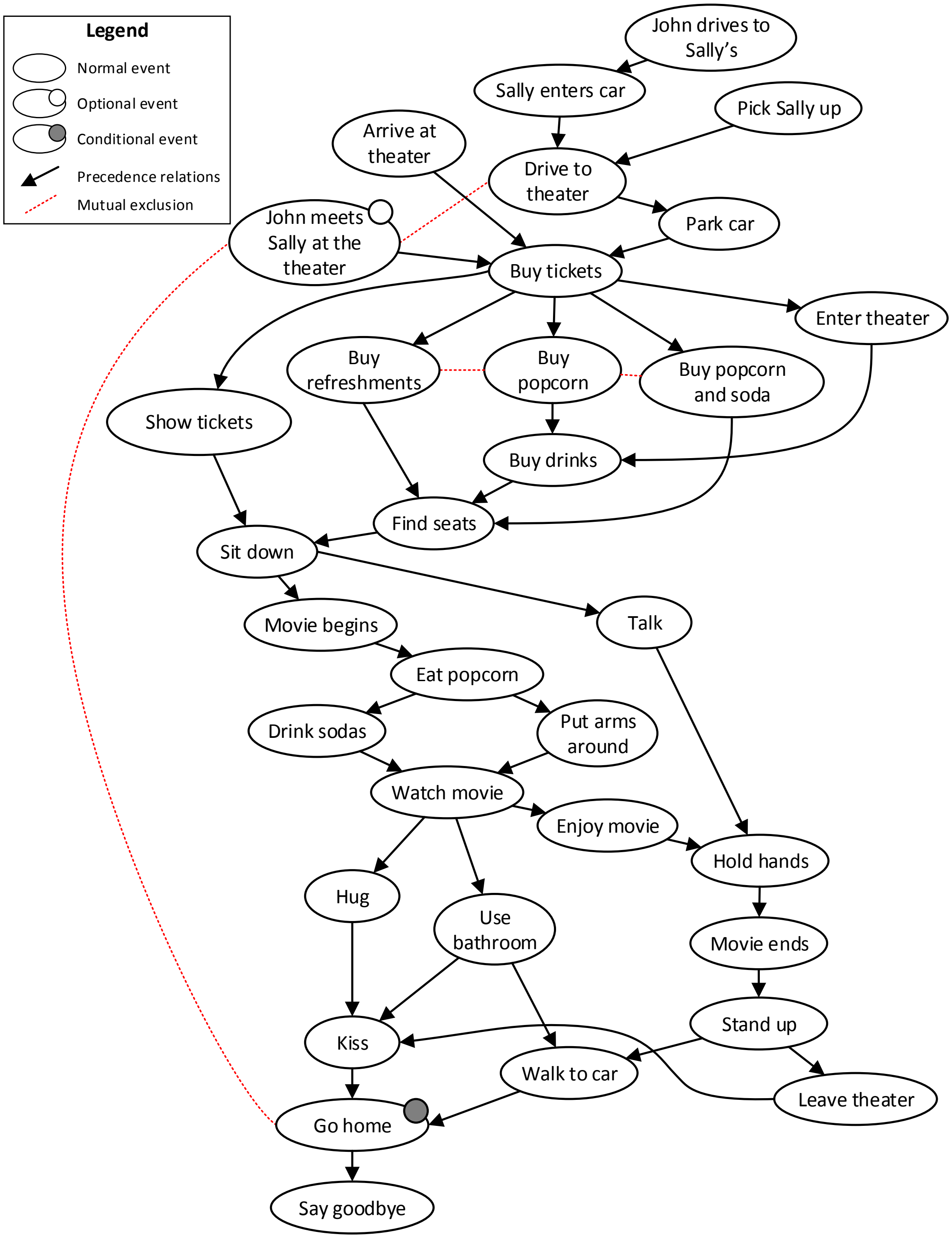


Figure 12: A plot graph for the movie date situation, created by the IQCP method.

more accurately. In general, we tend to see ordered relations when we expect causal necessity, and we see unordered events when ordering variations are supported by the data.

Comparing the two graphs, The IQCP graph seems to be more reasonable. For example, the IQCP graph positions the event “park car” correctly but the smart thresholding method does not. The IQCP graph also correctly positions “enjoy movie” between “movie begins” and “movie ends”, and positions “buy drinks” correctly before “find seats”. The ST graph correctly identifies that John and Sally could put their arms around each other most of the time during the movie, and they can drink the soda anytime, whereas the IQCP method restrains the two events “put arms around” and “drink sodas” more narrowly. However, the IQCP graph correctly identifies John and Sally can talk during most time of the movie, but the ST graph puts the “talk” event after “stand up” and before “kiss”. The ST graph also contains fewer events than the IQCP graph because events not temporally related to any other events are not included in the graph.

In general, the smart thresholding method errs on the side of omitting precedences with low probability, and the IQCP graph errs on the side of including too many precedences, so the ordering of events may be overly constrained. In the movie date situation, whereas the storyline is mostly linear without significant branches and contingencies, the IQCP method appears to work better.

The mutual exclusions detected are correct, but some are missing. The mutual exclusion between “John meets Sally at the theater” and “go home” is because when the two people meet at the theater, they usually say goodbye at the theater and the story ends. If John picked up Sally, however, human writers tend to describe how John and Sally go home together. We miss some mutual exclusion relations, such as the one between “John drives to Sally’s” and “pick Sally up” because they are two different descriptions of the same event. The missing mutual exclusions exist in both graphs, as they use the same mechanism for detecting mutual exclusion relations.

3.9.1 Methodology

We evaluate the two graphs in the movie date situation empirically with human judges recruited from Amazon Mechanical Turk. The learned precedence relations as well as the absence of such relations are checked. From the ST graph, we randomly sampled 30 pairs of adjacent events, i.e. events in the automatically generated plot graph that are ordered by a $before$ relation without any interstitial events. We also randomly sampled 29 pairs of parallel events, i.e. events for which the plot graph indicates no necessary ordering relative to one another. From AMT, we recruited 144 workers. Each worker was paid $0.12-0.20$ to check seven pairs of events.

Each worker was instructed to consider each pair of events in the context of going on a date to the move theater. Each pair of events (A, B) was presented to a worker in a randomized order (50% of workers saw A before B and 50% of workers saw the opposite) and workers were asked whether (a) it is more likely that A comes before B, (b) it is more likely that B comes before A, or (c) that they are unable to tell which should come first. In order to detect cheating or randomly clicking, two of the seven pairs were designed as validation questions. These two pairs of events do not appear anywhere in the plot graph, but were manually written and have obvious orderings. If a worker provided a wrong answer on either of two pairs, all of his or her answers were considered invalid and discarded. Each worker was allowed to participate in the study only once.

3.9.2 Results

With the gold standard created by the crowd workers, we evaluate the two methods for learning precedence relations. The results for the smart thresholding (ST) graph and the IQCP graph are shown in Table 5 and Table 6 respectively. The differences between the two tables, resulted from subtracting the ST results from the IQCP

Table 5: Accuracy of the learned precedence relations by the smart thresholding method

	entropy= 0		entropy < 0.4		entropy< 0.6		entropy< 0.8		entropy< ∞	
	acc.	% pairs	acc.	% pairs	acc.	% pairs	acc.	% pairs	acc.	% pairs
All	**0.76**	29	0.64	42	0.66	54	0.54	73	0.53	100
Adjacent	**1.00**	40	0.93	50	0.90	67	0.82	73	0.70	100
Parallel	0.20	17	0.20	34	0.25	41	0.22	79	**0.28**	100
All-sans-ends	**0.80**	24	0.73	37	0.68	46	0.48	76	0.49	100
Adjacent-sans-ends	**1.00**	35	0.50	50	0.83	60	0.69	80	0.60	100
Parallel-sans-ends	0.33	14	0.40	24	**0.43**	33	0.25	76	0.33	100

Table 6: Accuracy of the learned precedence relations by the IQCP method

	entropy= 0		entropy < 0.4		entropy< 0.6		entropy< 0.8		entropy$< \infty$	
	acc.	% pairs	acc.	% pairs	acc.	% pairs	acc.	% pairs	acc.	% pairs
All	**0.88**	29	0.76	42	0.72	54	0.64	73	0.57	100
Adjacent	**1.00**	37	0.94	47	0.88	60	0.79	76	0.70	100
Parallel	0.33	14	0.29	33	0.31	43	0.33	67	**0.34**	100
All-sans-ends	**0.91**	26	0.86	37	0.78	49	0.66	72	0.57	100
Adjacent-sans-ends	**1.00**	31	0.91	45	0.85	55	0.75	76	0.65	100
Parallel-sans-ends	0.50	14	**0.64**	21	0.53	36	0.44	64	0.40	100

Table 7: Differences in accuracy of the learned precedence relations by the two methods. The ST scores are subtracted from the QICP scores.

	entropy$= 0$	entropy < 0.4	entropy< 0.6	entropy< 0.8	entropy$< \infty$
All	0.12	0.12	0.06	0.1	0.04
Adjacent	0	0.01	-0.02	-0.03	0
Parallel	0.13	0.09	0.06	0.11	0.06
All-sans-ends	0.11	0.13	0.1	0.18	0.08
Adjacent-sans-ends	0	0.41	0.02	0.06	0.05
Parallel-sans-ends	0.17	0.24	0.1	0.19	0.07

results, are summarized in Table 7.

The rows in those tables indicate subsets of the data. The first three rows show the results from all sampled pairs, all sampled adjacent pairs, and all sampled parallel pairs (the remaining rows are explained later). The columns measure accuracy—the percentage of time human workers agree with our plot graphs—at different levels of worker agreement. We measure human agreement on each pair of events as the entropy of their answers. The entropy for the j^{th} pair of events (a_j, b_j) is defined as:

$$\text{entropy}(X_j) = -\sum_{i=1}^{3} P(x_{ji}) \ln P(x_{ji}) \tag{30}$$

where $x_{ji} \in \{\text{before}(a_j, b_j), \text{before}(b_j, a_j), \text{parallel}(a_j, b_j)\}$. The probability distribution $P(X_j)$ is observed directly from human responses for the pair (a_j, b_j). The columns of Table 5 show statistics for event pairs with increasing entropy from left to right (i.e. decreasing worker agreement). For example, the first column include only pairs where workers unanimously agree (entropy = 0), which are 29% of all pairs evaluated (row "All"), and of those 29%, workers agreed with the ordering in the ST graph 76% of the time. Lowering the entropy threshold filters out pairs of events with low agreement from consideration.

3.9.3 Discussion

We draw four sets of observations about our plot graph learning algorithm in the movie situation:

- **Overall accuracy**. The smart thresholding method yields an overall accuracy over 53%, whereas the IQCP method has a higher accuracy of 57%. Both are well above the purely random baseline of 33%. When we examine only pairs for which workers perfectly agree with each other our accuracy is as high as 76% for smart thresholding, and 88% for IQCP, although this only accounts for about 29% of our total sampled pairs. We found that when humans could not reach consensus on a pair of events, they tend to also disagree with our system. The IQCP method consistently outperforms the smart thresholding method in overall accuracy.

- **Adjacent events**. Our system is very accurate when it comes to determining when a before relation should exist between a pair of events. Workers agree with our before relations at about 90% of the time when they can reach good consensus (entropy < 0.6). This suggests our algorithm is a good model of the ground truth. Accuracy decreases as workers begin to disagree but remains high ($\approx$0.7-0.8) in the worst case. The IQCP method shows slightly lower performance in the conditions of entropy < 0.6 and entropy < 0.8, suggesting it recognizes precedence relations more aggressively than the other method.

- **Parallel events**. For all parallel pairs, workers agreed with our system only 28-33% of the time. However, workers agreement is generally lower for parallel events than adjacent events. Unanimous agreement can be reached on only 14-17% of all pairs, in contrast to 37-40% for adjacent pairs. The lack of agreement on many of these pairs suggests insufficient collective social expectation of the orderings. The reason that individual worker may prefer one ordering to another may be attributed to the way questions were asked (which ordering is more likely). Even though one ordering is likely, the other ordering may be also possible. Our results suggest that although we are missing before relations that would eliminate parallel events our system may be correctly placing events as parallel in the graph when there is very little agreement on ordering.

  The IQCP method generally performs better on parallel events than the smart thresholding method. As the IQCP method recognizes more precedences, it is more selective in recognizing the lack of precedence relations, and thus produces better results. However, the success of the IQCP method is also attributed to the fact that the movie date situation contains a mostly linear storyline, without significant branches and contingencies.

- **Removing events with sparse data**. During the crowdsourcing process, I observe that people start and end the exemplar stories at different points. Thus, data about events at the beginning and the end are more sparse than rest of the plot graph. It is reasonable to postulate that data sparsity has affected the accuracy of learned precedences at the two ends of the graph. To test this hypothesis, the last three rows of result tables show the results when we remove all pairs involving events before "buy tickets" in the ST graph and three events at the end: "go home", "walk to car", and "say goodbye" from the data. 43 pairs of events remain after these events are removed.

  The removal significantly improves the accuracy for the detection of parallel events, where improvements are seen in most conditions of human worker agreements. Accuracy of parallel events improves by 3% to 13% in the ST graph, and 6% to 35% in the IQCP graph. The overall accuracy improves when human workers reach unanimous agreement by 3% to 4%. We do not see improvements when human workers disagree with each other, but again it may be particularly difficult, even for humans, to determine the correct relations between those events.

  The results suggest the data sparsity is a real concern for graph learning, and there is room for the story acquisition procedure to address this issue. After the removal of these events, the IQCP method outperforms the smart thresholding method by an even higher margin, further suggesting the strength of the IQCP method.

Overall, this evaluation demonstrates the system is capable of learning plot graphs that is consistent with human intuition to an encouraging degree. When humans can reach an agreement, they usually agree with the system. Our results show our accuracy percentage reaches high 80s to low 90s when humans reach unanimous agreement among themselves. Even when we include all pairs of sentences where humans have little consensus among themselves, our accuracy is still around 60%, well above the purely random baseline of 33%. When humans cannot agree with each other, they tend not to agree with the system, but that is more or less expected. Scarcity of data, especially at the beginning of the story, can negatively affect the result of learning.

3.10 Limitations and Future Work

Many story generation and story understanding systems (e.g. [99, 128, 154, 201]) have demonstrated the utility of causal relations between events for tasks of Narrative Intelligence. However, in this dissertation, I do not discover causal relations from the exemplar stories. One difficulty I face in discovering causal relations is people tend to omit very obvious causes and effects when writing the exemplar stories, even when instructed not to do so. For example, in preliminary experiments people tend to mention either "finding a table" and "sitting down" at a restaurant, because the two actions are immediately causally related and almost always follow each other. Since we only observe second-hand information, the most obvious causal relations tend to be missing in the data and become difficult to detect[2].

Ideally, one can learn causal conditions so they can be used to create plan-like structures as in aforementioned Narrative Intelligence systems. However, the human notion of causality remains vague. Some theories [31, 125] postulate that causal relations can be captured by simple statistics, such as in the comparison between the probability of an event e given a cause c, $P(e|c)$, and the probability of the same event in the absence of a cause, $P(e|\neg c)$. Such causal relations may be learned by controlled and unbiased experiments of counterfactuals [134]. On the other hand,

[2] In the final experiment, we used those two events as an example in our instructions to workers, telling them not to omit events, but the effects were limited.

other theorists [79] argue causality is perceived as the abnormal necessary condition. For example, a train wreck has several necessary conditions, including that the training is traveling at high speed, that the rail is faulty, and that the train is heavy enough to break out of the rail. Most people would consider the faulty rail as the cause of the train wreck, as this condition deviates from what is believed as normal. This view suggests learning a normality model, as done in this dissertation, may facilitate the identification of causal relations.

The current model contains two types of relations: precedence and mutual exclusion. Although the two types can capture a large number of situational variations, it does distinguish between events that are happening simultaneously and events that should happen sequentially with an indeterminate ordering. For example, the two events “John and Amy lay in bed” and “John’s wife Sally opened the door” in the extra-marital affairs situation happen simultaneously. The two events “John paid for the food” and “The cashier passed John the food” in the fast food restaurant situation can happen in any order. The difference is that, in the latter case, the events must happen in an particular order, but this order can be arbitrary. The reason my approach cannot detect this difference is that I only observe exemplar stories that are linear sequences. Simultaneous events and events with no specific orderings all have to be linearized in the exemplar stories, and there is little observed information to differentiate the two types of relationships. Additional annotations may be required to learn simultaneity between events.

Clustering quality can further be improved by recruiting crowd workers to modify the automatically identified clusters. The system also cannot identify events that may happen in multiple locations and for more than once. These improvements are left for future work.

3.11 Summary

In this section, I have presented a formal definition of plot graphs, which extends traditional plot graphs used in interactive narrative systems, and includes events in a situation, precedence relations and mutual exclusions between events, and events deemed to be optional and conditional. I have also presented a crowdsourcing procedure for acquiring suitable exemplar stories and algorithms for learning each components of the plot graph from those exemplars.

The learned plot graphs have been evaluated in two ways: whether the learned events match manually created gold standards, and whether the learned precedence relations (or the lack of them) match the intuition of crowd workers recruited from Amazon Mechanical Turk. The evaluation shows the learning algorithms achieve satisfactory performance. The precedence learning algorithm achieves an accuracy percentage in high 80s and low 90s, when human judges can reach unanimous agreement.

However, the utility of learned plot graphs can only be tested by applying them in tasks of Narrative Intelligence. In the next two chapters, I will present methods for generating and understanding stories based on the learned graphs.

CHAPTER IV

GENERATING AND TELLING STORIES

> *No story comes from nowhere; new stories are born from old—it is the new combinations that make them new.*
>
> — Salman Rushdie

The utility of learned plot graphs can only be demonstrated by their use in supporting various Narrative Intelligence tasks. In this chapter, I demonstrate story generation and storytelling based on learned plot graphs. The pipelined generation process covers all three tiers in Bal's model [5]: fabula, sjuzhet, and text (or media).

The story generation/telling pipeline starts with the generation of a linearized fabula. As noted in the previous chapter, my representation does not distinguish between simultaneous events and sequential events whose order is determined at run time. Thus, we directly generate linearized fabulas whose events are linearly sequenced. This generation is described in Section 4.1. Afterwards, we select some events from the fabula in order to create interesting stories in Section 4.3. Finally, in Section 4.4, we tackle the problem of generating story text for a given sjuzhet by reusing crowdsourced sentences in the learned plot graphs. The generated fabula and story tests are evaluated by human judges recruited on Amazon Mechanical Turk. The evaluations are described in Sections 4.2 and 4.5 respectively.

4.1 Fabula Generation

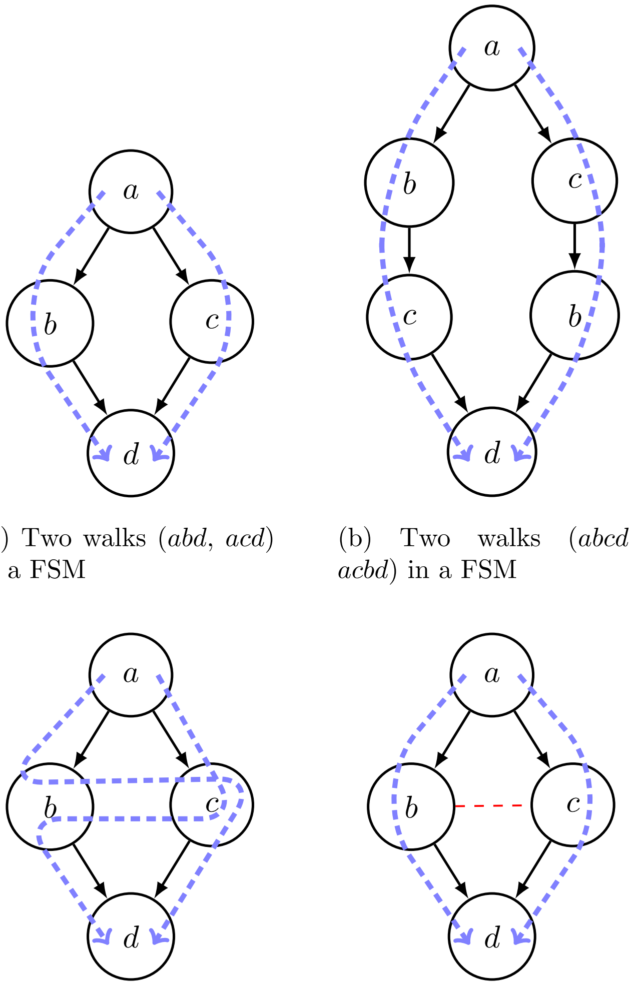
In this section, I describe how the Scheherazade system generates a linearized fabula from a plot graph. By showing what events may happen in in a social or procedural situation and how these events are involved in a number of precedence relations and mutual exclusion relations, a plot graph defines a space of linear event sequences. Each sequence can be considered as a linearized fabula.

4.1.1 Legal Passages Through a Plot Graph

In a finite-state machine (FSM) defined by a directed graph, a valid sequence of states is a walk through the graph that starts and ends in the correct states. A walk in a directed graph is defined as a sequence of vertices where vertices adjacent in the walk are also adjacent in the direct graph.

Unlike a finite-state machine, the directed edges or precedence relations in a plot graph define a partial order, rather than transitions, between the events. Therefore, in a plot graph, valid sequences of events are not walks. This difference is illustrated in Figure 13. Figure 13(a) shows a finite-state machine where we can visit state d from either state b or state c, so valid sequences include *abd* and *acd*. In a plot graph with the same layout (Figure 13(c)), however, we can only visit event d after both event b and c have been visited, creating two valid sequences *abcd* and *acbd*. The partial-order graph compactly captures variations in the fabula caused by events happening in different orders. To represent the two sequences in 13(c) as a FSM, we would need to replicate the vertices b and c, as illustrated in Figure 13(b). In contrast, we can easily represent the two sequences in Figure 13(a) as a partial plot graph by simply adding a mutual exclusion relation, as illustrated in Figure 13(d). The compactness of partial-order plot graph facilitates learning, as we do not need to replicate the learned events in the graph. However, this representation does make graph traversal more complex. In this section, I explain the procedures for traversing plot graphs.

I call a valid sequence of event through the plot graph a *passage*. Since each vertex in a plot graph represents an event, we refer to the action of visiting a vertex as *executing* an event. To generate valid fabula is to generate passages through a plot



(a) Two walks (*abd*, *acd*) in a FSM

(b) Two walks (*abcd*, *acbd*) in a FSM

(c) Two passages (*abcd*, *acbd*)

(d) Two passages (*abd*, *acd*)

Figure 13: Contrasting passages in plot graphs with walks in finite-state machines. The circles are vertices, and black solid arrows are directed edges. Red dashed lines are mutual exclusion relations. The valid walks and passages are shown as dashed blue arrows.

graph.

A valid passage is defined by the following rules that determine when an event can be executed.

1. Incoming precedence relations to an event are treated as necessary conditions for the execution of that event. Therefore, an event e can be executed if and only if (1) all of its parents meet any one of the four conditions:

    - the event has no parents.
    - the parent has been executed.
    - the parent is optional.
    - the parent has been removed from the graph by a mutual exclusion relation.

    and (2) that none of e's children has been executed, as explained below.

2. Optional events may be skipped and are not considered to be necessary conditions of any events. However, once an optional event e has been skipped and one of its children has been executed, event e cannot no longer be executed. This is because executing event e after one of its children violates a precedence relation. Those events are called *expired events*.

3. When an event e is executed, by the definition of mutual exclusion relations, all events that are mutually exclusive to e are removed from the plot graph. Removed events cannot be executed.

4. Event removals by mutual exclusions are recursive. If all parents of an event e have been removed from the plot graph because of mutual exclusion relations, event e must also be removed.

    As noted previously, mutual exclusion relations may be inadvertently omitted due to the application of a universal threshold T_m. This recursive deletion propagates mutual exclusion from parents to children and compensates for omitted mutual exclusion relations. This recursive removal continues until no events in the plot graph meet the criterion for removal.

The recursive deletion is equivalent to adding mutual exclusion relations in the plot graph. We determine the events being deleted at run time instead of adding the relations when learning the plot graphs. This is because it is not always clear what mutual exclusion relation should be added to the plot graph, due to problems such as race conditions. Section 5.3 describes these problems in details.

5. If an event e has more than one parents, and some, but not all, of the parents have been removed, event e will not be removed from the plot graph. In this case, parents of the removed events become parents of event e in order to avoid losing structural information.

6. When an optional or conditional event e is removed by a mutual exclusion relation, its children will not be removed. Parents of event e will become direct parents of event e's children.

7. We stop executing events when no events may be executed, or when we reach one ending event in the plot graph. An ending event is an event with no children.

The algorithms for maintaining the legality of passages are shown as Algorithm 3 and Algorithm 4. By implementing rule 1, the ExecutableEvents function finds the set of all executable events at any time. The set of all executable events at any time is termed the *fringe*. The passage generation algorithm (which I will explain in the next section) executes one vertex e_{step} at one time. After e_{step} is added to the passage, the UpdateGraph function is called to bring the graph up to date. The ExcludeEvents function computes events that should be removed by mutual exclusion relations and their transitive closure. These events are directly removed from the graph and will no longer be considered. To preserve structural information,

Algorithm 3 Graph Maintenance 1

```
function ExecutableEvents($G = \langle E, P, M_x, E_o, E_c \rangle$, history, $E_{expired}$)
    $fringe \leftarrow \emptyset$
    for each event $e \in E, e \notin E_{expired}$ do
        if $\forall (e^p, e) \in P, e^p \in history \vee e^p \in E_o$ then
            $fringe \leftarrow fringe \cup \{e\}$
        end if
    end for
end function

function UpdateGraph($G = \langle E, P, M_x, E_o, E_c \rangle$, history, $e_{step}$)
    $E_{expired} \leftarrow$ ExpiredEvents($G, e_{step}$, history)
    $E_{excluded} \leftarrow$ ExcludedEvents($G, e_{step}$)
    AddLinksAcross($G, E_{excluded}$)
    $E \leftarrow E \setminus (E_{excluded} \cup E_{expired})$
    return $G$
end function

procedure AddLinksAcross($G = \langle E, P, M_x, E_o, E_c \rangle, E_x$)
    for each event $e \in E_x$, if $e \notin E_o \wedge e \notin E_c$ do
        for each $(e, e^c) \in P$ do                 ▷ Add links between all predecessors
            for each $(e^p, e) \in P$ do                    ▷ of $e$ and successors of $e$
                $P = P \cup \{(e^p, e^c)\}$
            end for
        end for
    end for
end procedure
```

according to the generation rule 5, parents of removed events are linked to the children of removed events with precedence relations, as performed by the AddLinksAcross function.

4.1.2 Generating Passages With a Search Algorithm

The fabula generation algorithm executes one event (or one vertex) at a time. During this process, we add the executed events to the generated passage. Algorithm 5 shows the fabula generation algorithm. In order to implement rule 2 and 6, the GenerateFabula function first adds precedence relations from parents of optional events to children of optional events, and precedence relations from parents of conditional events to children of conditional events. This ensures the removal of any optional

Algorithm 4 Graph Maintenance 2

```
function ExcludedEvents($G = \langle E, P, M_x, E_o, E_c \rangle, e_{step} \in E$)
    $E_{excl} \leftarrow \emptyset$
    for each mutual exclusion $\langle e_{step}, e_x \rangle$ in $M_x$ do
        $E_{excl} \leftarrow E_{excl} \cup e_x$
    end for
    for each mutual exclusion $\langle e_x, e_{step} \rangle$ in $M_x$ do
        $E_{excl} \leftarrow E_{excl} \cup e_x$
    end for
    repeat
        $E_{old} \leftarrow E_{excl}$
        for each event $e \in E$ do
            $parents \leftarrow$ direct_parents($e$)
            if $parents \subset E_{excl}$ then          ▷ if all direct parents have been excluded
                $E_{excl} \leftarrow E_{excl} \cup e$                    ▷ then exclude $e$ as well
            end if
        end for
    until $E_{excl} == E_{old}$                              ▷ repeats until convergence
    return $E_{excl}$
end function
function ExpiredEvents($G = \langle E, P, M_x, E_o, E_c \rangle, e_{step} \subseteq E$, history)
    $E_{expr} \leftarrow \emptyset$
    for each temporal order $\langle e, e_{step} \rangle$ in $P$ do
        if $e \notin history$ then                        ▷ $e_i$ has not been executed, but
            $E_{expr} \leftarrow E_{expr} \cup e$      ▷ executing it now will violate a temporal order
        end if
    end for
end function
```

or conditional events by mutual exclusion relations will not recursively remove any children. The function then generates n fabulas stochastically and selects the best according to a given fitness function in EvaluateFitness.

The PassThruGraph function is responsible for generating one legal passage through the plot graph. Selection from the fringe is performed by the function PickEvent. The selection could be deterministic or stochastic. When the heuristic is stochastic, it selects events from a distribution over the fringe, which is the case shown in Algorithm 5. Taken together, the stochastic heuristic can provide a local estimate of which event might lead to a good fabula (however goodness is defined), whereas

Algorithm 5 Fabula Generation

```
function GenerateFabula($G = \langle E, P, M_x, E_o, E_c \rangle$)
    AddLinksAcross($G, E_o$)
    AddLinksAcross($G, E_c$)
    $fabula \leftarrow \emptyset$
    $best \leftarrow -\infty$
    for $i = 1 \rightarrow n$ do                      ▷ Find the best of the $n$ stories, only if
        $new\text{-}fabula \leftarrow$ WalkGraph(G)          ▷ events are stochastically selected
        $value \leftarrow$ EvaluateFitness($new\text{-}story$)
        if $value > best$ then
            $fabula \leftarrow new\text{-}fabula, best \leftarrow value$
        end if
    end for
    return $fabula$
end function

function PassThruGraph($G = \langle E, P, M_x, E_o, E_c \rangle$)
    $fabula \leftarrow \langle\rangle$
    while not IsCompleteStory($G, fabula$) do
        $fringe \leftarrow$ ExecutableEvents($G, fabula$)
        $e \leftarrow$ PickEvent($fringe$)        ▷ Select an event $e$ according to a heuristic
        $fabula \leftarrow fabula + e$                     ▷ Append event $e$ to the story
        $G \leftarrow$ UpdateGraph(G, fabula)
    end while
    return $fabula$
end function

function IsCompleteStory($G = \langle E, P, M_x, E_o, E_c \rangle, history \subseteq E$)
    $E_{end} \leftarrow \emptyset$                                   ▷ find ending events
    for each event $e$ in $E$ do
        if $\nexists e^s \in E, \langle e, e^s \rangle \in T$ then
            $E_{end} \leftarrow E_{end} \cup e$          ▷ Events without successors are ending events
        end if
    end for
    return ($history \cap E_{end} \neq \emptyset$)
end function
```

the fitness function (i.e. EvaluateFitness) provides a global evaluation of the entire fabula. The stochastic method of generation is computationally less efficient, but may be convenient if it is difficult to provide an estimate of story quality based on only part of the fabula. On the other hand, if we employ a deterministic mechanism for event selection, it suffices to generate one story. If the plot graph is completely correct, every legal event sequence should be coherent, but other aesthetics measures, such as novelty, may differ for different event sequences.

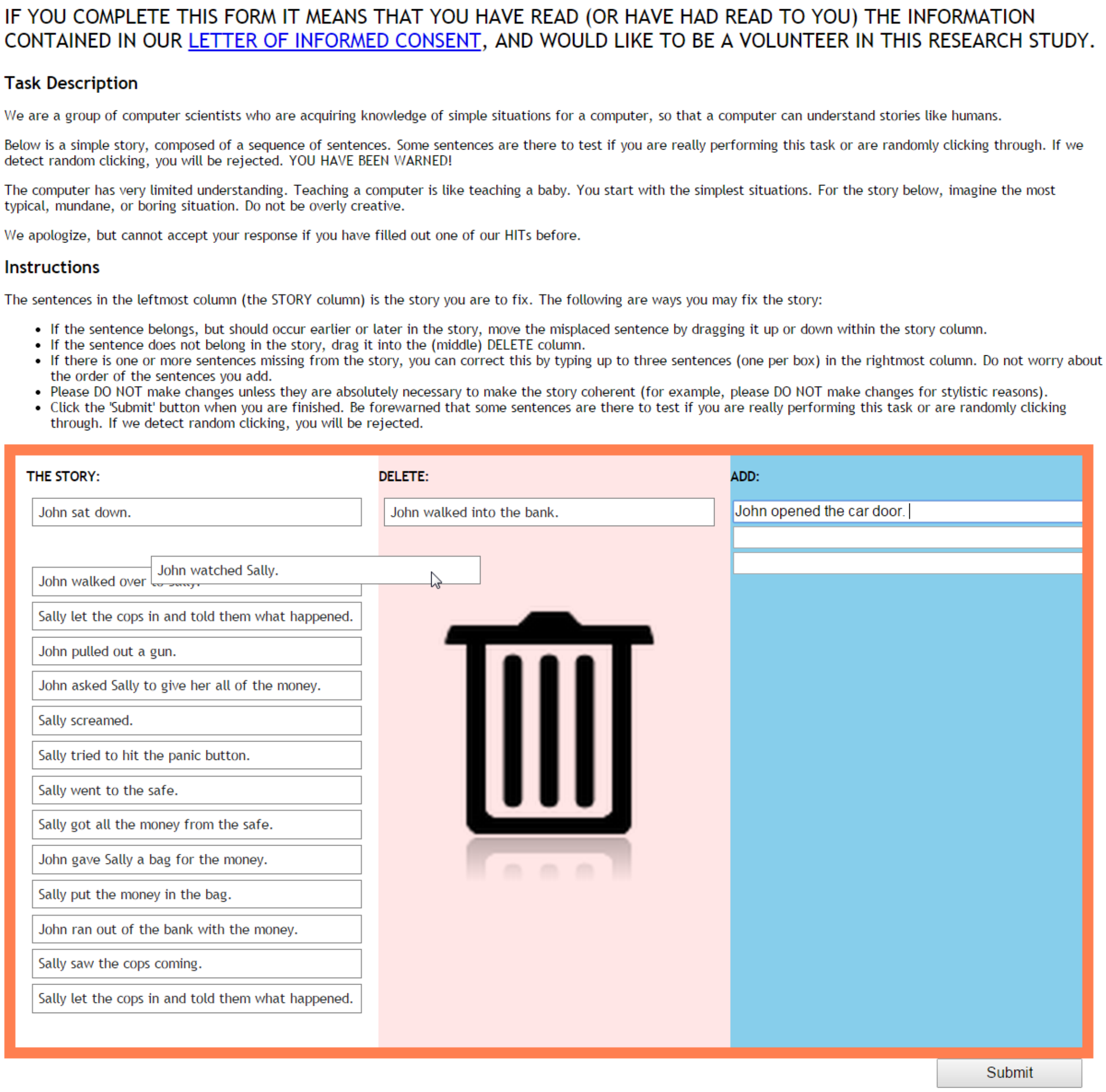
The generation algorithm produces over a million different legal passages through the plot graph for the bank robbery situation (Figure 15). The authorial leverage [30]—the ratio of possible narratives to authoring effort—of our system is high considering the input contains only 60 examplar stories. The quality of generated fabulas is assessed in Section 4.2.

4.2 Evaluating the Generated Fabula

This section describes a large-scale evaluation of the coherence of fabula generated from the bank robbery plot graph as shown in Figure 15. As all legal event sequences should be coherent in a completely correct plot graph, this evaluation provides a measure for the quality of the learned plot graph (including precedence, mutual exclusion, and optional events) in addition to the quality of the generation procedure.

4.2.1 Methodology

This evaluation focuses on the coherence of generated fabula. Although coherence is arguably the most important measure, it is certainly not the only possible quality measure for a story. In a traditional approach for story generation, a knowledge engineer may carefully design the knowledge to guarantee the coherence of the story. In a machine learning approach, however, I do not have much freedom to adjust the learned representation. Thus, it is important to test if it is possible to create coherent stories from the learned representation.



IF YOU COMPLETE THIS FORM IT MEANS THAT YOU HAVE READ (OR HAVE HAD READ TO YOU) THE INFORMATION CONTAINED IN OUR LETTER OF INFORMED CONSENT, AND WOULD LIKE TO BE A VOLUNTEER IN THIS RESEARCH STUDY.

Task Description

We are a group of computer scientists who are acquiring knowledge of simple situations for a computer, so that a computer can understand stories like humans.

Below is a simple story, composed of a sequence of sentences. Some sentences are there to test if you are really performing this task or are randomly clicking through. If we detect random clicking, you will be rejected. YOU HAVE BEEN WARNED!

The computer has very limited understanding. Teaching a computer is like teaching a baby. You start with the simplest situations. For the story below, imagine the most typical, mundane, or boring situation. Do not be overly creative.

We apologize, but cannot accept your response if you have filled out one of our HITs before.

Instructions

The sentences in the leftmost column (the STORY column) is the story you are to fix. The following are ways you may fix the story:

- If the sentence belongs, but should occur earlier or later in the story, move the misplaced sentence by dragging it up or down within the story column.
- If the sentence does not belong in the story, drag it into the (middle) DELETE column.
- If there is one or more sentences missing from the story, you can correct this by typing up to three sentences (one per box) in the rightmost column. Do not worry about the order of the sentences you add.
- Please DO NOT make changes unless they are absolutely necessary to make the story coherent (for example, please DO NOT make changes for stylistic reasons).
- Click the 'Submit' button when you are finished. Be forewarned that some sentences are there to test if you are really performing this task or are randomly clicking through. If we detect random clicking, you will be rejected.

THE STORY:

John sat down.

John watched Sally.

John walked over to Sally.

Sally let the cops in and told them what happened.

John pulled out a gun.

John asked Sally to give her all of the money.

Sally screamed.

Sally tried to hit the panic button.

Sally went to the safe.

Sally got all the money from the safe.

John gave Sally a bag for the money.

Sally put the money in the bag.

John ran out of the bank with the money.

Sally saw the cops coming.

Sally let the cops in and told them what happened.

DELETE:

John walked into the bank.

ADD:

John opened the car door.

Figure 14: The user interface that human judges used to edit fabula generated by the system.

Although the notion of coherence can be subjective, I attempt to measure it quantitatively as the minimum number of edits that human judges need in order to make a fabula coherent. Human judges read the fabulas as a list of sentences and make changes to render the stories more coherent. Judges could delete or reorder using a drag-and-drop interface. They can also write up to three new events to be inserted into the story, but could not specify the location of insertion. To detect random clicking and cheating, one obviously incorrect event was inserted in every fabula. A judge's response is only accepted if that event is correctly deleted. The web interface used by the human judges is shown in Figure 14.

450 people were recruited on AMT and paid 20-30 cents as compensation. 60 participants edited the 60 stories from the crowdsourced corpus. 100 additional stories were uniformly sampled from all possible fabulas that could be generated by the plot graph. No fitness functions to filter the generated fabulas were used. 300 participants edited generated fabulas such that three judges saw each fabula.

To establish a baseline, another 30 fabulas were generated by uniformly sampling from all events in the plot graph. That is, no learned graph structures, including any precedence relations, mutual exclusion relations or optionality of events were used. Therefore, these 30 fabulas were most likely to be illegal according to the plot graph. The length of the random fabulas were set to 23, which is approximately the average length of fabulas generated from the plot graph. Each of the 30 fabulas were edited by 3 human workers, resulting in 90 edited fabulas. For all computer-generated fabulas, the most frequent sentence of the underlying natural language cluster was selected to describe each event in the text presented to the human workers.

The number of added events and deleted events from the original fabula presented to the human worker can be easily obtained from data. However, obtaining the number of movements is a little more complex, as counting the number of mouse clicks or drag-and-drops will inevitably include human operations that were performed by

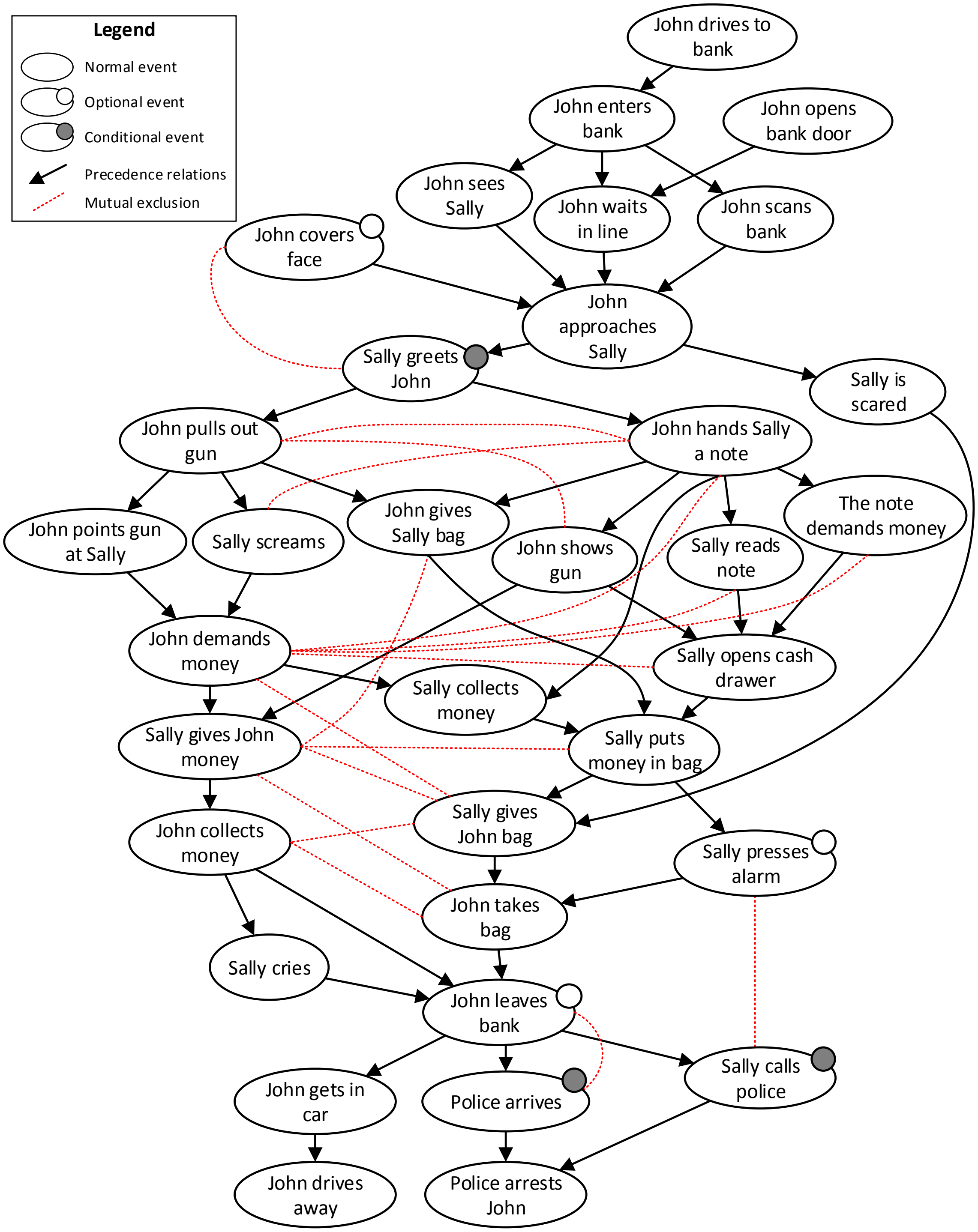


Figure 15: A plot graph for the bank robbery situation, created by the smart thresholding method. $T_p = 0.75, T_m = 0.05$

mistake and later undone. To avoid this problem, an algorithm is used to determine the number of movements required to change the original event sequence to the edited sequence submitted by a human worker. The algorithm computes the number of inverted pairwise orderings in the original event sequence associated with each event. It then picks the event with most inverted pairwise orderings, places it in a position that minimizes inverted orderings, and repeats until the two sequences become identical. The number of movements performed by this algorithm is recorded.

4.2.2 Results

As each edit suggests a problem with the fabula, less edits indicate better quality. The number of events added, deleted, or moved are shown in Table 8. Average numbers of additions, deletions, and reorderings after normalizing for story length are also shown. The Welsh t-test was used to determine if the difference between human-authored and computer generated fabulas is statistically significant at $p < 0.05$. In the Computer column, I denote whether the differences between the human-written fabulas and the computer-generated fabulas are statistically significant. In the Random column, the first symbol (i.e. before the comma) denotes statistical significance between the random fabulas and the human-written stories, and the second symbol denotes statistical significance between the random fabulas and the computer-generated fabulas.

I find pronounced differences between the random fabulas and other two conditions. For most measures, the differences are strongly statistically significant with the only exception in the number of added events after normalizing for story length. On average 2.29 events were deleted from the random fabulas, which is three times the number of deleted events in computer-generated fabulas. The number of movements for the random fabulas is also roughly three times as the AI condition.

Compared to the random baseline, for most edit metrics, the human and the AI conditions appear to be similar. No significant differences exist in number of

Table 8: Statistics of edits made by human judges to the generated fabulas. * indicates the difference between the two columns is statistically significant. § indicates the difference between the two columns is not statistically significant.

	Human	**Computer**	**Random**
Mean original length	12.78	23.14 *	23.00
Mean final length	11.82	21.89 *	20.71
Mean events added	0.33	0.49 §	0.80 *,*
Mean events added (normalized)	0.028	0.021 §	0.035 §,*
Mean events deleted	0.30	0.76 *	2.29 *,*
Mean events deleted (normalized)	0.02	0.03 §	0.10 *,*
Mean events deleted (2 events withheld)	0.28	0.27 §	-
Mean events deleted (2 withheld, norm.)	0.02	0.01 §	-
Mean events moved	0.57	4.88 *	14.67 *,*
Mean events moved (normalized)	0.04	0.21 *	0.64 *,*

added events between the two conditions. After normalizing for length, the computer condition appears to have less added events than the human condition. The exception is in the number of moved events, where the differences across all three conditions are large and statistically significant. However, some reorderings could be consistent with the plot graph, as discussed in the next section.

I find a small but statistically significant difference in the number of events deleted between computer-generated fabulas and human-written fabulas. Although statistically significant, the mean difference between conditions is less than half of an event. The significance vanishes when two events "wait in line" and "open door" are withheld. The two events account for 64.5% of deletes, and occurred rarely in the corpus (6 and 9 times out of 60), which explains why the system has less certainty about their inclusion. After normalizing for length and withholding, the AI has less deletions than human authors.

There is a clear and statistically significant difference in the length of the stories for the two conditions. The computer generated stories contains on average 23.14 sentences, which is almost twice as long as the human-written story. This is because the learning algorithm of plot graphs merge events from many exemplar stories, so the plot graph contains more events than any single exemplar stories. I will discuss and address this issue in Section 4.3.

4.2.3 Discussion

With the help of the random baseline, I conclude that this novel method of measuring story coherence can indeed capture differences in story coherence. The number of movements appear to be the most sensitive measures. In comparison to deletion and movements, the number of added events is less sensitive, although is still reflective of the overall coherence. There are several possible reasons for this observation. First, it could be due to the fact that the user interface allows easy drag-and-drop operations for re-ordering and deletion, but no drag-and-drop for adding events. Second, only three blanks were provided for addition, which restricted editing but also prevented human workers from deviating from the typical story. Third, as the random fabulas were fairly long with 23 events, the human workers could have found the events they needed and did not need to add new events.

I conclude that the story generation algorithm does not omit essential events any more often than human authors. I conclude that despite the existence of multiple incompatible alternatives throughout the plot graph, the system was able to differentiate between them and does not add events that contradict human intuition. This result is attributed to the use of mutual exclusion relations to successfully separate incompatible events.

There are several reasons why a judge may have reordered events. The fact that these events are not deleted indicates that they contribute to the story but are not ideally positioned. The precedence relations may be under- or over-constraining the events, leading to incoherent fabula. Alternatively, I also find some edits not to be strictly necessary for story coherence, indicating judges may prefer one ordering to another out of aesthetic concerns rather than strictly coherence reasons.

Overall, we find 32.3% of moves to be consistent with the plot graph, indicating that the reordering exists in another legal story and the occurrence of under-constraining. The rest of the moves violate temporal constraints in the plot graph, indicating events being over-constrained. However, the changes may be due to aesthetic rather than coherence reasons. For example, "pressing the alarm" is over-constrained to occur in the second half of the story. Two moved events account for a plurality of inconsistencies: "get in car" is uniformly moved from the end to the beginning, but both positions seem reasonable; "Sally cries" is a rare event in the corpus. Removing these two events reduces the inconsistencies to 44% of generated stories.

Leslie Kaelbling (personal communication) proposed a hypothesis that if a story is excessively incoherent, human judges may find it difficult to make changes at all. However, our observations suggest the human judges tend to make more changes than necessary. This may be explained by the special incentive provided by the AMT, where the employer has total control over whether to pay a worker or not. Thus, workers may be motivated to do a little more work than they consider to be bare minimum, just in order to appeal to the employer.

In conclusion, the fabulas generated from learned plot graph is largely on par with human-written stories in the aspect of added and removed events. The plot graph likely contains some errors that over- or under-constraints the ordering of events, although we should not interpret all reorderings as errors. To my knowledge, this is a first study showing computer-generated stories may match human-written stories in some ways and is thus very encouraging. Considering the fact that these stories were generated as random passages through the plot graph, this indicates the structures, including the precedence relations, mutual exclusion relations, and optional/conditional events, in learned plot graphs are of high accuracy and quality.

4.3 Sjuzhet Generation

In Section 4.2, we have seen that the generated fabula are usually longer than human-written stories. The reason that humans do not include every event in their story is probably that some events are too obvious or too mundane to be told. From the events being told, a human reader is able to infer that those untold events have happened. For example, if a storyteller says she found a seat in the movie theater, in the lack of contradictory evidence, the audience would assume that she sat down. This inference process is critical for story understanding [72, 207], and successful storytellers know how to make use of this inference to their advantage. However, SCHEHERAZADE learns scripts that include most events in the situation. By following the scripts, the system tends to generates overly verbose stories.

Filtering out some of the less interesting events in the fabula is a commonly used sjuzhet technique. Narratologists Barthes [8] and Chatman [29] both noted that events in a narrative carry different importance. Barthes used the term *nuclei* to describe important, plot-driving events, and the term *catelyzers* to describe unimportant events adding to the aesthetics of the story. Chatman used the term *kernels* and *satellites* to refer to the same concepts. The ability to differentiate important events from others has been considered as an integral part of Narrative Intelligence. For example, a comprehensive questionnaire for measuring narrative abilities of children developed by Heilmann *et al.* [78] includes the evaluation if the child includes critical events and deemphasizes minor events. Cheong and Young [34] use the kernel/satellite differentiation to manipulate sjuzhets and generate suspenseful stories.

In this work, I generate sjuzhets consisting of both kernel events and satellite events. Following Bathes and Chatman, I postulate that an effective storytelling strategy is to tell a set of kernel events that establish the situation and major storyline, plus a set of interesting satellite events. Kernel events set up the situation and form a major causal chain in the story. They create a context without which the story would become difficult to understand. Example of kernel events are "we went to a restaurant", "we ordered steaks" and "we finished the meal". On the other hand, we can create interesting satellite events as infrequent and atypical events in a situation, such as "the waiter complained the tip was too little". Therefore, the central issue in sjuzhet generation becomes the computation of how typical an event is to a particular situation. My strategy for generate interesting sjuzhet selects the k most typical and k least typical events to produce a story of $2k$ length.

I introduce an algorithm called EVENTRANK, which determines the typicality of events in a plot graph, taking into consideration of the size of the event cluster, the structure of temporal orderings, as well as mutual exclusion relations. The algorithm is inspired by Personalized PageRank [76], which computes the importance of vertices contained in a strongly connected directed graph structure, as explained in the next two sections.

4.3.1 PageRank and Stationary Distributions of Digraphs

The PageRank algorithm [18] was developed to measure the importance of webpages on the Internet, which is modeled as a directed graph with webpages and hyperlinks.

Imagine a random algorithm that visits one webpage at one time epoch and uniformly randomly follows hyperlinks on the current page. The webpages then form a Markov chain, where the webpage visited in the current epoch depends only on the page visited in the previous epoch. In a network of n webpages, the webpage visited at time epoch t is a categorical distribution, parameterized by a n-dimensional vector $\boldsymbol{x}^{(t)}$. $x_i^{(t)}$ is the probability that we visit page i at time t. By measure theory, we require that the sum of all entries of $\boldsymbol{x}^{(t)}$ equal to 1 for all t.

$$\sum_{i=1}^{N} x_i^{(t)} = 1, \forall t \tag{31}$$

The change in the page being visited over time can be captured by a transition matrix A, whose entry A_{ij} denotes the probability of transiting from webpage j to webpage i. Again, the probability of transiting from webpage j to all webpages must be equal to 1.

$$\sum_{i=1}^{N} A_{ij} = 1, \forall j \tag{32}$$

Given no prior preferences for any single outgoing link to others, we can assign equal weight to each outgoing link.

$$A_{ij} = \begin{cases} 1/d_j & \text{if there is a directed edge from vertex } j \text{ to vertex } i \\ 0 & \text{if there is no directed edges from vertex } j \text{ to vertex } i \end{cases} \tag{33}$$

where d_j is the out-degree of vertex j. It is obvious that the webpage being visited at the next time epoch can be computed as multiplication by A.

$$\boldsymbol{x}^{(t+1)} = A\boldsymbol{x}^{(t)} \tag{34}$$

If the graph represented by A is irreducible and aperiodic, it can be shown that as t approaches infinity, the distribution of the current webpage will approach a distribution $\boldsymbol{x}^{(\infty)}$, regardless of her starting position $\boldsymbol{x}^{(0)}$.

$$\boldsymbol{x}^{(\infty)} = \lim_{c \to \infty} A^c \boldsymbol{x}^{(0)}, \forall \boldsymbol{x}^{(0)} \tag{35}$$

$\boldsymbol{x}^{(\infty)}$ is called the equilibrium distribution of the Markov chain formed by the webpages. In an irreducible graph, we can find a path from any vertex to any other vertex. In an aperiodic graph, we return to the same vertex at irregular intervals. For a formal discussion of stationary distributions of Markov Chains, the reader is referred to textbooks on Markov Chain Monte Carlo such as [158].

Given A, $\boldsymbol{x}^{(\infty)}$ can be easily computed as the eigenvector of A corresponding to the eigenvalue 1, which is also the biggest eigenvalue A. Thus, $\boldsymbol{x}^{(\infty)}$ is a property of the network structure captured by transition matrix A and is independent of the initial distribution $\boldsymbol{x}_0$. Intuitively, when we have randomly wandered in the network for sufficient time, where we started off would have little bearing on our current position. PageRank takes $\boldsymbol{x}_i^{(\infty)}$ to be the importance of the i^{th} webpage.

It is important to maintain the property of irreducibility and aperiodic. For example, if the network graph is not strongly connected and contains several components, we will not reach one component from the other. Thus, where we start becomes important. If the graph is periodic, we will always return to the same webpage after a period T, so where we start will also matter. To avoid these situations, PageRank allows random jumps from any page to every other page with a small probability. That is, we add another uniform matrix B to A.

$$\tilde{A} = \lambda A + (1 - \lambda)B \tag{36}$$

where

$$B_{ij} = \frac{1}{n} \tag{37}$$

and take the eigenvector of $\tilde{A}$, which is guaranteed to have a unique equilibrium distribution. The constant λ is usually set to 0.85.

The Personalized PageRank [76] is a variant of PageRank that utilizes a simple insight: The added matrix B (in Equation 37) does not have to be uniform. We can bias the matrix and increase the importance of certain webpages, in order to incorporate information other than the graph structure. The next section describes how we can incorporate extra information of event frequency and mutual exclusion relationships to determine the typicality of events in a situation.

4.3.2 The EventRank Algorithm

The precedence relations and vertices in a plot graph form a directed acyclic graph, which is not strongly connected. To satisify the requirement of strong connectivity, precedence relations are added between each ending event (events with no outgoing links) and each starting event (events with no incoming links). The transition matrix A can then be constructed based on the precedence relations according to Equation 33.

In addition to the precedence relations, we want to consider two pieces of information when computing the typicality of events in a situation. The first is the frequency of events being mentioned in crowdsourced exemplar stories, since one story typically only mentions a subset of all events. Events that are mentioned more often are probably more typical to a situation. Second, mutual exclusion represents alternative branches in a plot graph, and they should have effects on typicality values. Specifically, we would like mutual exclusive events to weaken each other, and symmetrical branches to have similar typicality values. The information is incorporated into the PageRank algorithm by modifying the matrix B.

As we favor events that are mentioned more frequently in the exemplar stories, when events i and j are not mutually exclusive, we let

$$B_{ij} \propto f_i \tag{38}$$

where f_j is the frequency of event i being mentioned in the exemplar stories. If events i and j are mutually exclusive, however, we penalize event i (and by symmetry, event j) by letting

$$B_{ij} \propto f_i - \tau o_j \tag{39}$$

where

$$o_i = \sum_{k \notin X_i} \frac{f_k}{\text{card}(X_i)} \tag{40}$$

and X_i is the set of vertices mutually exclusive to vertex i, and $\text{card}(X_i)$ is its cardinality. τ is a value in $[0, 1)$, and I typically set it to between 0.3 and 0.5. The rationale behind the above formula is that if an event e_i has fewer mutually exclusive events, it is more likely to be included in a story and hence more powerful in weakening other events. The power of weakening is evenly distributed among all events mutually exclusive to e_i.

Finally, we normalize B so that each column sum up to 1. We again compute the stationary distribution $\boldsymbol{x}^{(\infty)}$ by finding the eigenvector with the eigenvalue 1 of the matrix $\tilde{A} = \lambda A + (1 - \lambda) B$. Figure 16 shows the bank robbery plot graph with the typicality values of each event. λ is set to 0.7 and τ set to 0.5. The structure of the graph is generated by the smart thresholding method. For the purpose of visualization, the median of the typicality values has been normalized to 1.

We can make two observations from the diagram. First, structural features of the graph are reflected by the typicality values. For example, vertices that multiple paths converge onto, such as "John approaches Sally", and "John leaves bank" are considered to be typical. Second, comparable alternative events also receive similar typicality values. The vertices "John pulls out gun" and "John hands Sally a note" denote two major situational variations and receive similar values. The two alternative endings "John drives away" and "Police arrests John" also receive similar values. Our observation suggests the EventRank algorithm does capture the structural information in plot graphs, including both the precedence relations and mutual exclusion relations.

4.3.3 Generating Different Sjuzhets

Table 9 shows events in a fabula and the corresponding typicality values, generated from the bank robbery plot graph. Employing the typicality values in the selection of events, we can create different sjuzhets. For example, we can create a short summary by picking some of the most typical events from a fabula. Shown in bold in Table 9, the top five most typical events in the bank robbery situation are “John approaches

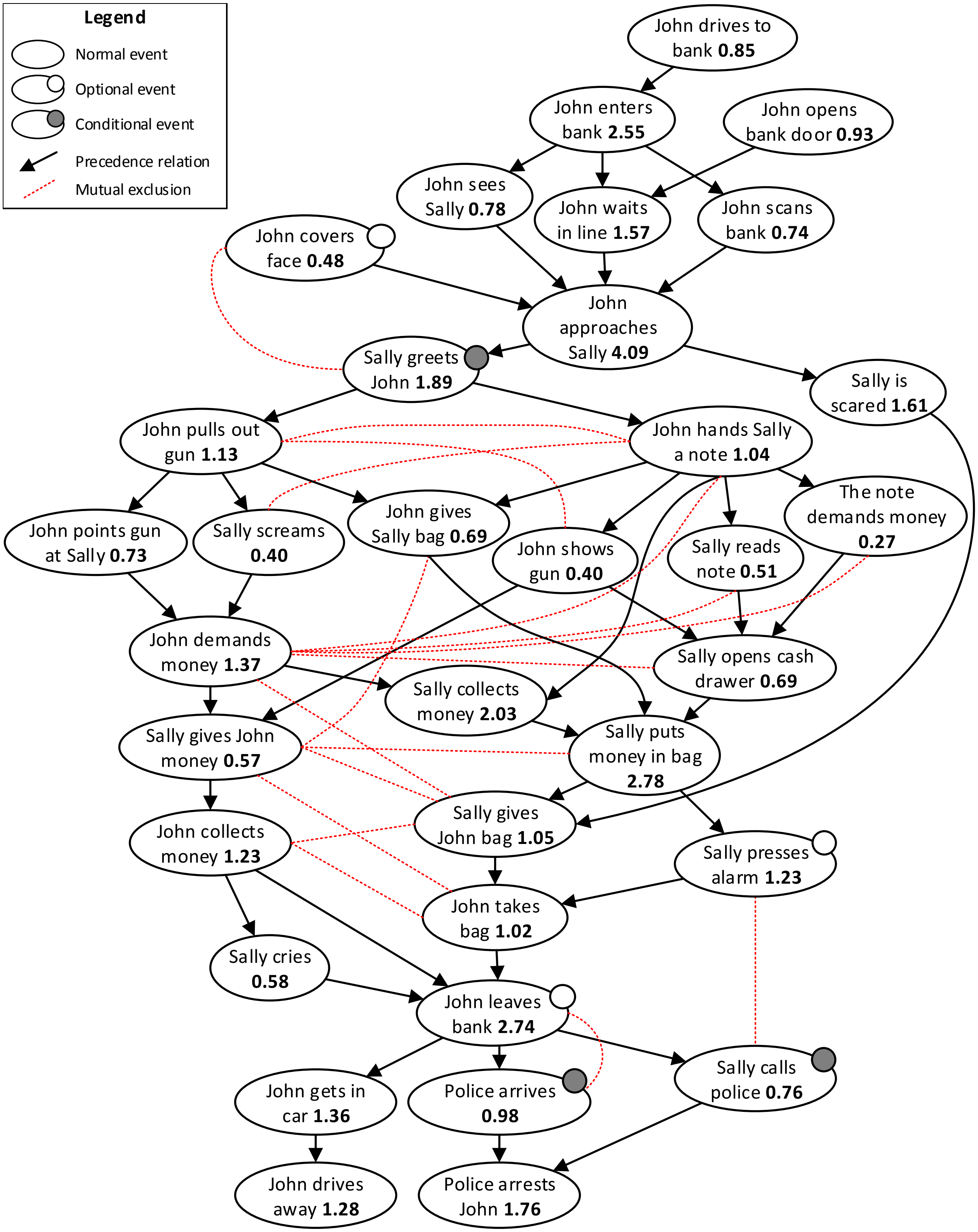


Figure 16: The typicality of events in the bank robbery situation. τ is set to 0.5.

Table 9: Selecting events by typicality from a fabula to create sjuzhets. The five most typical events are shown in bold, and the five least typical events are underlined.

Event	**Typicality**
John drives to bank	0.85
John opens bank door	0.93
John enters bank	2.55
John scans bank	0.74
John waits in line	1.57
John sees Sally	0.78
John approaches Sally	4.09
Sally greets John	1.89
John pulls out gun	1.13
John points gun at Sally	0.73
Sally screams	0.4
Sally is scared	1.61
John demands money	1.37
John gives Sally bag	0.69
Sally collects money	2.03
Sally puts money in bag	2.78
John collects money	1.23
Sally presses alarm	1.23
John leaves bank	2.74
Sally cries	0.58
Police arrests John	1.76

Sally", "Sally puts money in bag", "John leaves bank", "John enters bank", "Sally collects money". Ordering the five events in the order they appear in the fabula, they form a mostly intelligible summary of the bank robbery situation. The summary does not mention John demanding money or collecting money, but those might be inferred.

In the terminology of Chatman [29], the most typical events can be considered as kernel events, and the least typical events as satellite events. Therefore, we may make the sjuzhet more interesting sjuzhet by adding some of the least typical events to the most typical events. The most typical events, or the kernels, establish the social situation, so the readers can understand the context. The least typical events, or the satellites, make the story more interesting. The top 5 least typical events in this fabula include: “Sally screams”, “Sally cries”, “John points gun at Sally”, “John scans bank”, and “John drives to bank”, which are underlined in Table 9. Take the summary as the baseline, adding the top 5 least events to the top 5 most typical events reveals emotional states of Sally and arguably makes a more interesting sjuzhet.

4.4 Textual Realization

The final step of story generation is to describe the events in a selected medium. In this dissertation, the medium of choice is text. This section discusses how to generate the textual realization of distinct styles for a given sjuzhet by selecting sentences from event clusters. There are several motivations for generating text with different styles. First, a narrative may be told to achieve different communicative goals. For example, if a storyteller wants to cheer up her audience, she may employ a positive tone. In contrast, the telling of horror stories may benefit from a negative storytelling style making more use of words inciting fear and suspense. Second, a narrative can be told by different narrators in different styles. Consider the application of virtual characters. Depending on the circumstances, mood and personality, a virtual character may choose to speak very succinctly, or to be very talkative. The technique of focalization utilizes multiple viewpoints in telling one narrative (see Section 2.2.2 for a review). Therefore, developing parameterized methods for telling stories in different styles can provide practical benefits.

In this section, I first describe an approach for crowdsourcing interesting sentences. After that, I discuss criteria for selecting textual description of individual events, followed by an Viterbi-style algorithm that aligns adjacent sentences to improve textual coherence.

4.4.1 Textual Interestingness

We investigate two aspects of language that affect the interestingness of generated stories. The first is the amount of details provided, and the second is the degree that the story language resembles the language used in fictions.

We first model the amount of details as the probability of a sentence in English, as Information Theory suggests a less likely sentence should contain more information and therefore more details. We compute the probability of an English word as its frequency in the Google N-Gram corpus. Due to the large size of the corpus, these frequencies approximate word probabilities in general English. We compute the probability of a sentence using the bag-of-word model where each word is independently drawn from a multinomial distribution. Thus, the probability of sentence S containing words $w_1, w_2, \ldots, w_k$ each appearing $x_1, x_2, \ldots, x_k$ times is

$$P(S) = \frac{\left(\sum_{i=1}^{k} x_i\right)!}{\prod_{i=1}^{k}(x_i!\,)} \prod_{i=1}^{k} P(w_i)^{x_i} \tag{41}$$

where $P(w_i)$ is the probability of the word w_i. In this experiment, I used the average frequency over the 10-year period of 1991 to 2000 in the “English 2012” corpus to compute $P(w_i)$. Stop words are removed before computation.

We further consider the style of language is as how much it resembles fictional novels. The language used in fictions has distinctive word choice as fictions tend to accurately describe actions (e.g. “snatch” instead of “take”), emotions, and make less use of formal words (e.g. “facility”, “presentation”). If a word appears more often in fiction books than in all books, we can presume that its use may create a sense that a story is being told in a literary manner. Therefore, the fictionality of a word w is the ratio

$$f_w = P_{\text{fic}}(w)/P(w) \tag{42}$$

where $P(w)$ is the probability of a word computed previously and $P_{\text{fic}}(w)$ is the probabilities of a world appearing in the “English Fiction 2012” corpus from the

Table 10: Fictionality of example words computed from the Gooogle N-Gram corpus

Word	POS	Fictionality
goodbye	NN	11.24
sugary	JJ	10.04
hobbit	NN	8.31
laughing	NN	8.19
softly	RB	7.02
feature	NN	0.23
menu	NN	0.23
presentation	NN	0.23
restoration	NN	0.23
produce	VB	0.20
appropriate	JJ	0.20

Google N-Gram corpus. See Table 10 for ficitionality for some example words. Words with low fictionality, such as "feature" and "restoration", tend to appear in technical or business documents.

The fictionality of a sentence is aggregated from fictionality values of individual words as an exponentiated average:

$$\mathrm{fic}(S) = \frac{\sum_{w \in W} \exp(\alpha f_w)}{\mathrm{card}(W)} \tag{43}$$

where W is the set of words in sentence S, and $\mathrm{card}(W)$ is its cardinality. α is a scaling parameter. The exponential function puts more weights on more fictional words so that a few highly fictional words are not canceled off by many words with low fictionality.

Table 11 shows some results of our heuristics for determining the interestingness of sentences. We observe that the most probable sentence (MostProb) usually provides a concise summary for the event. The most fictional (MostFic) sentence usually contains more subjective emotions and character intentions, as indicated by the words "smirk", "nervously" and so on. The least probable (LeastProb) sentence is usually longer and contains more objective details.

Table 11: Sentences selected from event clusters using the probability criterion, the fictionality criterion, and their harmonic mean as the MID criterion.

Example event 1: John covers face

- MostProb: John put on a fake mustache.
- LeastProb: John kept his head down as he pulled open the outer door and slipped his Obama mask over his face.
- MostFic: MF: John looked at his reflection in the glass of the door, gave himself a little smirk and covered his face.
- MID: John kept his head down as he pulled open the outer door and slipped his Obama mask over his face.

Example event 2: Sally puts money in bag

- MostProb: Sally put $1,000,000 in a bag.
- LeastProb: Sally put the money in the bag, and collected the money from the 2 tellers next to her.
- MostFic: Sally quickly and nervously stuffed the money into the bag.
- MID: Sally quickly and nervously stuffed the money into the bag.

Example event 3: John drives away

- MostProb: John drove away.
- LeastProb: John pulled out of the parking lot and accelerated, thinking over which route would make it easier to evade any police cars that might come along.
- MostFic: John sped away, hoping to get distance between him and the cops.
- MID: John sped away, hoping to get distance between him and the cops.

As a balance between the most fictional and the least probable, we combine the two selection criteria and create the heuristic for the most interesting details (MID) by using the harmonic mean. We first rank each sentence under the least probable and the most fictional criteria: r_{LP} and r_{MF}. That is, the least probable sentence has $r_{\text{LP}} = 1$ and so on. The mean rank is:

$$r_{\text{MID}} = \frac{2\, r_{\text{LP}}\, r_{\text{MF}}}{r_{\text{LP}} + r_{\text{MF}}} \tag{44}$$

The sentence with the lowest r_{MID} is picked as the sentence with the most interesting details.

4.4.2 Smooth SentiWordNet

A story may be told in positive or negative tones in order to achieve different communication goals, or to tell the story from the perspective of characters with different mood. For instance, a positive tone may be used to cheer up the audience, whereas a negative tone may be suitable for telling horror stories.

To detect sentiments of natural language, in this section I describe a corpus-based technique for detecting the sentiment of English words and sentences. The technique builds off SentiWordNet [51], which tags each synset (word sense) in WordNet [116] with three values: positivity, negativity, and objectiveness, the three summing to 1. SentiWordNet was created by propagating known sentiments of a few seed words along word relationships in WordNet to provide good coverage of words. While this automatic approach creates good coverage, I also find that it produces many erroneous values, resulting in unreliable sentiment judgments.

I propose an unsupervised, corpus-based technique to correct errors found in the original library and expand its coverage beyond words appearing in WordNet. The underlying intuition is that words in the same neighborhood, including adjacent words and words in the same sentences and the same paragraph, should share similar sentiments, allowing us to automatically "smooth" any errors in the original sentiment library. In addition, words closer should have a stronger influence than words farther away.

A review of similar lexicon expansion techniques is provided in Section 2.3. In comparison to existing techniques, my method makes use of full narrative text to consider long-distance influences and considers the fact that word association tends to diminish as we move farther away from a word. The use of a fiction-only corpus produces a narrative-specific lexicon, as we know sentiments can change depending

Table 12: Examples of fictional books obtained from Project Gutenberg

Gutenberg ID	Book Title
11	Alice's Adventures in Wonderland
12	Through the Looking-Glass
15	Moby Dick
16	Peter Pan
24	O Pioneers!
27	Far from the Madding Crowd
32	Herland
33	The Scarlet Letter
35	The Time Machine
36	The War of the Worlds
41	The Legend of Sleepy Hollow
42	The Strange Case of Dr. Jekyll and Mr. Hyde
44	The Song of the Lark
45	Anne of Green Gables
46	A Christmas Carol in Prose; Being a Ghost Story of Christmas
47	Anne of Avonlea
51	Anne of the Island
54	The Marvelous Land of Oz
55	The Wonderful Wizard of Oz

on the context [108].

We obtained 9108 English books from Project Gutenberg (http://www.gutenberg.org) that are labeled as fiction. Some of the book titles are shown in Table 12. The complete list of books can be downloaded at http://boyangli.co/SSWN/booklist.txt. These books are tagged with parts of speech (POS) and lemmatized with the Stanford POS Tagger [186]. Each pair of lemma and POS is considered a unique word. For every occurrence of a target word we want to compute sentiment value for, we consider a neighborhood of 100 words, i.e. 50 to the left and the right of the target word. The target word is at position 0. The words to its immediate left and right are at position -1 and 1, and so forth. Only nouns, verbs, adjectives and adverbs in complete sentences in a neighborhood can influence the target word. The index set W includes their position. For a word w_i at position $i \in W$, we place a Gaussian kernel function g_i centered at its position, which indicates the influence of word w_i on a word at position j:

$$g_i(j) = \exp\left(\frac{-(i-j)^2}{d}\right) \tag{45}$$

where the parameter d determines how fast the function diminishes with distance, and is empirically set to 32. In the k^{th} neighborhood, the sentiment $s_{w_0}^k$ of the target word is computed as a weighted average of all kernel functions at position 0:

$$s_{w_0}^k = \frac{\sum_{i \in W_k} s_{w_i}^{\text{swn}} g_i(0)}{\sum_{i \in W_k} g_i(0)} \tag{46}$$

where $s_{w_i}^{swn}$ is the sentiment retrieved from SentiWordNet, i.e. the difference between the positive and negative sentiments for w_i. The SentiWordNet value for any word has no influence on itself, i.e. $0 \notin W_k, \forall k$. The final sentiment value for the target word s_{w_0} is the average of all its occurrences in the corpus.

$$s_{w_0} = \frac{\sum_{i=1\ldots K} s_{w_0}^k}{K} \tag{47}$$

We aggregate sentiments of individual words in sentence S, again using the exponential average:

$$\text{fic}(S) = \frac{\sum_{w \in U} \text{sign}\,(f_w) \exp\,(\beta |f_w|)}{\text{card}(U)} \tag{48}$$

where card(U) is the cardinality of set U, which contains any noun, verb, adjective or adverb in that sentence. β is a scaling parameter. The exponential function ensures that words expressing strong sentiments are weighted more heavily than words with weak sentiments.

Using this corpus-based technique, a new sentiment dictionary is constructed, which I call Smooth SentiWordNet (SSWN). I selected a subset of English words that are of interests to the storytelling task to be included in the dictionary. The exemplar stories in two previously crowdsourced social situations—dating at the movie theater

Table 13: Some most positive and most negative words in Smooth SentiWordNet

Word	POS	Sentiment
bone-chilling	JJ	-8.004
shriek	NN	-6.738
scop	VB	-6.734
wail	VB	-6.301
panic	VB	-5.679
sob	NN	-5.642
scream	NN	-5.316
nightmare	NN	-4.954
terror	NN	-4.261
panic	NN	-4.048
weep	VB	-3.840
enjoyable	JJ	1.821
deal	NN	1.974
beam	VB	1.984
admire	VB	2.005
immensely	RB	2.028
sexy	JJ	2.139
captivate	VB	2.393
small-framed	JJ	2.992
nacho	NN	3.149
snack	VB	3.277
tryed (sic.)	JJ	4.621

and bank robbery—contain 1001 unique nouns, verbs, adverbs and adjectives. From the corpus of fiction books, the aggregated influences of each adjectives and adverbs on their neighbors are measured. Highly influential adjectives and adverbs were added to the dictionary, producing a total of 7559 words. After computing the raw sentiment values for these words, we normalize the values so that 1 percentile and 99 percentile of the values fall in the range of $[-1, 1]$, to account for outliers. Twenty-two most positive and negative words in SSWN are shown in Table 13. For the two hundred most positive and negative words, See Appendix A. The entire dictionary can be

Table 14: Example sentences selected from event clusters with positive and negative sentiments

Example event 1: Sally puts money in bag

- Positive: Sally continued to cooperate, putting the money into the bag as ordered.
- Negative: Sally's hands were trembling as she put the money in the bag.

Example event 2: Sally cries

- Positive: Sally cried, somewhat relieved it may be over soon.
- Negative: Sally felt tears streaming down her face as she let out sorrowful sobs.

Example event 3: Sally calls police

- Positive: Sally described John as best as she could to the police.
- Negative: Still shaken, Sally reached for the phone and in a panicked manner called the police.

Example event 4: John opens bank door

- Positive: John took a deep breath and opened the bank door, letting an elderly woman exit before he entered himself.
- Negative: John opened the bank door while his heart was beating fast.

Example event 5: John pulls out gun

- Positive: John pulled out the gun, still smiling.
- Negative: John reached behind his back and withdrew his pistol.

downloaded at http://boyangli.co/SSWN/dictionary.txt.

We can observe that most of these words are put into the correct category. The order of negative words are also fairly accurate, e.g. "wail" is more negative than "sob" and "weep". The positive words contain a few more misclassifications such as "small-framed" and "beam" than negative words. Noises resulted from typos in the original text (e.g. "tryed") and errors in lemmatization (e.g. "captivate" was probably lemmatized from "captivating"). Overall, the most positive and most negative words seem qualitatively largely accurate.

Table 14 shows some of the most positive and most negative sentences. We find the results to reflect the valences of individual words. In example events 1-3, individual words like trembling or relieve dominate the entire sentence, and we can correctly identify positive and negative sentences. In example event 4, elderly and woman have positive valences, which coincide with the semantic meaning of the sentence. However, there are also cases where the aggregation of individual words valences deviates from the semantic meaning of the sentence. In example 5, the positive value of smile is the main reason for selecting the positive sentence, but smiling criminals may appear even scarier than usual. Due to language phenomena such as sarcasm and readers' personal judgments, the aggregated sentiment for a sentence is not 100% accurate. I describe a human study that quantitatively evaluates Smooth SentiWordNet in Section 4.5.3.

4.4.3 Connecting Sentences

For each event, we can find individual sentences ranked highest for any criterion or combinations of criteria using the harmonic mean. However, this selection does not consider the coherence between sentences and may results in incoherent texts due to two major problems: (1) previously mentioned objects can suddenly disappear and previously unmentioned objects can appear, and (2) a sentence can repeat actions in the previous sentence.

To address these problems, we propose a Viterbi-style algorithm, which considers both selection criteria for individual sentences and the interconnection between sentences. In a hidden Markov model, the Viterbi algorithm generates a sequence of hidden variables that best explains a sequence of observed random variables. The algorithm relies on two things specified in a hidden Markov model: One, the probabilities of a hidden variable generating any observation. That is, the observation indicates preference over the values of a hidden variable. Two, the probabilities of any hidden variable transiting to the hidden variable in the next time slice. That is,

Algorithm 6 A Viterbi-Style Algorithm for Generating Story Text

```
function GenerateText(event sequence $\langle c_1, c_2, \ldots, c_n \rangle$)
    for each sentence $s_k \in \{s_1, s_2, \ldots, s_m\}$ in event cluster $c_n$ do
        $(seq_k, \text{score}(seq_k)) \leftarrow$ BestSeqEndingIn$(s_k, c_n)$
    end for
    return the highest scored sequence from $seq_1, seq_2, \ldots, seq_m$
end function

function BestSeqEndingIn$(s_i, c_j)$
    for each sentence $s_p \in \{s_1, s_2, \ldots, s_m\}$ in event cluster $c_{j-1}$ do
        $(seq_p, \text{score}(seq_p)) \leftarrow$ BestSeqEndingIn$(s_p, c_{j-1})$        ▷ stored previously
        $new\_seq_p \leftarrow seq_p + s_i$
        $\text{score}(new\_seq_q) \leftarrow \text{score}(seq_q) + \text{score}(s_q, s_i) + \text{score}(s_i)$
    end for
    $best\_seq \leftarrow$ the highest scored sequence from $new\_seq_1, \ldots, new\_seq_m$
    return $(best\_seq, \text{score}(best\_seq))$
end function
```

we have preferences over pairs of values for adjacent variables.

Our problem is similar as we want to find the highest scored sentence sequence based on preferences over sentences in each event cluster, as well as preferences on how adjacent sentences connect. We do not consider connection between non-adjacent sentences. Specifically, we score the connection between any two sentences s_i, s_j as

$$\text{score}(s_i, s_j) = \log \frac{\text{shared\_nouns}(i, j) + 1}{\text{shared\_verbs}(i, j) + 1} \tag{49}$$

where shared_nouns(i, j) is the number of nouns shared by the two sentences, and shared_verbs(i, j) is the number of verbs shared by the two sentences. Similarly, we score individual sentences as the reciprocal of their ranks according to any selection criterion c:

$$\text{score}(s_i) = \frac{1}{\text{rank}_c(s_i)} \tag{50}$$

Our algorithm is shown as Algorithm 6. The BestSeqEndingIn function is recursive, because in order to find the best sequence ending in a given sentence s_i^j from the j^{th} event cluster c_j, we need to consider the scores of best sequences ending in every sentence from the previous cluster c_{j-1}, in addition to the connection between every sentence from cluster c_{j-1} and s_i^j. Due to the Markov property, we do not need to consider previous clusters $c_1, \ldots, c_{j-2}$. We can then iterate over every sentence from cluster c_j to find the best sequence ending in cluster c_j. A dynamic programming approach can be used to store every sequence ending in every sentence from every cluster and their scores.

Computing the time and space complexity is straightforward. Suppose we have a sequence of n clusters and m sentences in each cluster. We first need to compute the score of each sentence, which takes $O(mn)$ time. In addition, we need to consider the connection between the m sentences in one cluster and the m sentences in the next cluster, which takes $O(m^2)$ time. This computation is repeated for the $n-1$ connections. Therefore, the total time complexity is $O(mn)+O(m^2(n-1)) = O(m^2n)$. At each iteration, we need to store the best sequence ending in each of the m sentences in the current cluster. Each sequences contains at most n sentences, one from each cluster. Therefore, the total space complexity is $O(mn)$.

4.5 Evaluating the Generated Story Texts

In this section, I present user studies aimed to evaluate the stories generated using different narrator styles and sentiments. The user study also evaluates the accuracy of Smooth SentiWordNet.

4.5.1 Crowdsourcing Colorful Textual Descriptions

Before generating stories, I performed a second round of crowdsourcing as an attempt to collect interesting event descriptions for each learned event cluster. In Section 3.2, I presented the protocol for crowdsourcing exemplar stories from AMT. For the purpose of learning plot graphs, crowd workers are instructed to write stories in simple and bland language. Though simplified language facilitates plot graph learning by sidestepping many hard natural language processing problems, it is not conducive to generating vivid or sentimental speech. Therefore, I need to perform a second round

Table 15: Statistics of the additionally crowdsourced stories to enhance the interestingness of storytelling

	Movie Date	Bank Robbery
# Stories	20	10
# Sentences	470	210
# Words per sentence	14.53	13.7
# Verbs per sentence	2.36	2.6

of crowdsourcing to acquire sentences that would interest human readers.

Once again, I recruited crowd workers on Amazon Mechanical Turk. Each worker was shown the events that constitute a complete story to help them understand the story context. For the compensation of $1, they are asked to write detailed descriptions for each of these events. They were cued, but not required, to describe characters' intentions, facial expressions and actions. All answers that describe the correct events were accepted. No filtering based on subjective judgment of the interestingness of the answers were performed. The crowdsourcing was performed for the movie date situation and the bank robbery situation.

Some statistics of these newly crowdsourced stories are shown in Table 15. The statistics show that the newly crowdsourced sentences are longer and use more verbs than the sentences in the original corpora intended to simplify learning. Some of these sentences are shown in Table 11 and Table 14. In Table 11, the least probable and most fictional sentences are mostly from this data set, whereas the most probably sentence typically comes from the original, simplified language exemplars.

4.5.2 Evaluating Generated Story Texts

With the newly crowdsouced colorful sentences added to the event clusters, I now describe an human study that quantitatively evaluates whether the narration styles I defined is consistent with human readers' intuition.

4.5.2.1 Methodology

A total of 52 undergraduate, masters, and doctoral students participated in our study. Each participants read two groups of stories in both the movie date situation and the bank robbery situation.

The two groups of stories were generated with the Viterbi-style algorithm with different sentence selection criteria. The newly crowdsourced sentences were added to the event clusters. As revealed in the previous study on story coherence (Section 4.2), Scheherazade may generate stories with coherence issues such as incorrectly ordered events. In order to avoid the influence of story coherence, I manually edited the event sequence to maintain its coherence. Discourse planning was not used to avoid confounding factors. All stories used in this user study can be found in Appendix B.

The first group of stories includes stories generated from the the most interesting details (MID) criterion, the most probable (MostProb) criterion, and a story where we use the MID criterion but penalize long sentences. After reading the stories, participants are asked to select the most interesting story, the most detailed story and the most concise story. Our hypothesis is that human readers will select the MID story as containing the most details and the most interesting, and the MostProb story as the most concise. We set α to 12. The third story, where we use the MID criterion but penalize long sentences, is used as a control condition. The least probable criterion tends to favor long sentences. One hypothesis is that by penalizing long sentences, we could describe similar amount of details in fewer words. The third story is added to test this alternative hypothesis.

The second group of stories includes a story with the most positive sentiments, and a story with the most negative sentiments. We set β to 16 and 2 for the movie data and bank robbery situation respectively. After reading the second group, participants are asked to select a positive and a negative story. We hypothesize human readers will agree with the algorithm's sentiment judgments.

Table 16: Accuracy of the detected story-level textual interestingness, conciseness, and sentiments. § denotes $p < 0.0001$. * denotes $p < 0.0005$.

	Participant Agreement %	
Test	Movie Date	Bank Robbery
Most Concise Story	90.38§	75.00*
Most Detailed Story	97.92§	100.00§
Most Interesting Story	88.46§	80.77§
Positive/Negative Stories	86.54§	92.31§

4.5.2.2 Results

Table 16 shows the percentage of human participants that agree with the algorithm. All results are predominantly positive and consistent with my initial hypotheses. In the movie date situation, the participants agreements are in upper 80s and 90s. In the bank robbery situation, there is lower agreement on which story is most concise, but unanimous agreement on which story is the most detailed.

The hypotheses are tested using a one-tailed hypothesis testing based on the multinomial/binomial distribution. For the three stories in group 1, the null hypothesis that human judges select the stories randomly suggests that the participants agreement should be close to 1/3. The second group contains only two stories, so the random baseline is 1/2. The statistical tests find we can reject the null hypotheses at very high confidence levels of $p < 0.0001$ and $p < 0.0005$. The alternative hypothesis that penalizing for length may maintain the level of interestingness has not been supported.

4.5.2.3 Discussion

In conclusion, the heuristics I developed for creating diverse narrator styles were shown to accurately capture the human intuition of interestingness, conciseness, and positive versus negative sentiments. The learning of narrator styles from large data sets can be considered successful.

However, it is arguably easier to detect the sentiment of an entire story than to detect the sentiment of individual sentences, because a few sentences labeled with wrong sentiments, when mixed together with many correctly labeled sentences in a long story, may be overlooked by the participants in the study. To further evaluate the constructed sentiment lexicon, Smooth SentiWordNet (SSWN), I also conducted a sentence-level study, as detailed in the next section.

4.5.3 Evaluating Smooth SentiWordNet

This user study aims to evaluate whether I can predict the sentiments of individual sentences using the sentiment lexicon Smooth SentiWordNet.

4.5.3.1 Methodology

From 45 event clusters taken from both situations, we first compute the top 3 most positive sentence and top 3 most negative sentences. Each participant saw a pair of positive and negative sentence randomly selected from the top 3, and was asked to choose the positive and negative sentence. A total of 52 undergraduate, masters, and doctoral students participated in our study. They performed 4678 comparisons of 265 unique pairs of sentences. SSWN labels a sentence as positive if it has higher sentiment than the median sentence in a cluster, and negative if it is lower.

The participants' responses created a gold standard for the sentences we selected. Based on these labels, I compared SSWN with the original SentiWordNet lexicon as well as the the sentiment detection technique by Socher *et al.* [169] from Stanford University. To compare with SentiWordNet, the word sentiment values in Equation 48 were substituted with values directly taken from SentiWordNet. A positive or negative label can then be produced for each sentence. I tuned β to maximize performance. To compare with Socher *et al.*'s method, I input the sentences into their web demo and took the overall label. The results are summarized in Table 17.

Table 17: Accuracy of the detected sentence-level sentiment, with comparison to SentiWordNet and the technique by Socher *et al.*. § denotes $p < 0.0001$.

	Participant Agreement %		
Test	Smooth SWN	SentiWordNet	Socher et al.
Sentence Sentiments (Individual)	70.76	59.60§	35.91§
Sentence Sentiments (Majority)	80.75	64.53§	39.25§

4.5.3.2 Results

Results are shown as Table 17. Overall, 70.76% of participants' decisions agree with the results produced by SSWN. The majority opinion on each pairs of sentences agree with our algorithm for 80.75% of the time. In addition, SSWN outperforms SentiWordNet by a margin of 11.16% to 16.22%, and outperform Socher *et al.*'s technique by 34.85% to 41.5%, but it is worth noting Socher *et al.*'s algorithm targets movie reviews and has not been tuned on our data set. A Chi-Square test shows the difference between the two conditions to be extremely statistically significant at the level of 0.0001.

4.5.3.3 Discussion

Results of this experiment indicates that SSWN can predict the overall sentiment of sentences much more accurately than the original SentiWordNet. The results strongly suggest that the corpus-based technique used to create SSWN is able to correct errors in SentiWordNet. The automatic method used by SentiWordNet propagates sentiment values along relations between words in WordNet, whereas my technique propagates sentiment values along neighborhoods in texts in a selected corpus. These two techniques can complement each other in creating a lexicon of high pratical utility.

4.6 Limitations and Future Work

While there are many sjuzhet creation techniques, in this dissertation I only consider one of the most important techniques: the ability to omit certain events in order to improve the interestingness of generated stories. I have not addressed the question whether the ordering of events could be re-arranged to achieve certain aesthetic effects such as suspense.

When creating an interesting sjuzhet, I consider how typical an event is to a situation. Typical events are used to establish a situation. However, the proposed EventRank algorithm does not consider if an event appears in many different situations. If an event is typical to many different situations, it may not be able to correctly establish the intended situation. A probabilistic formulation may differentiate $P(\text{event}|\text{situation})$ and $P(\text{situation}|\text{event})$. The evaluation of the EventRank algorithm is future work.

I proposed an algorithm for textual coherence based on noun agreements and verb differentiation. Text generation using more sophisticated coherence measures such as that proposed by Barzilay and Lapata [9] is also left for future work.

4.7 Summary

In this section, I present algorithms for generating stories based on learned plot graphs. The ability to generate stories and tell stories demonstrates the utility of the learned plot graphs.

Following the pipelined three-tier narrative model, the Scheherazade system sequentially generates fabulas, sjuzhets, and natural language texts of a story. I first define several graph-walking rules for generating fabulas, and the EventRank algorithm for evaluating the typicality of an event in a situation. After that, I propose heuristics for generating different storytelling styles by selecting sentences from crowdsourced sentences and build a sentiment dictionary, Smooth SentiWordNet, based on a large corpus of fiction books. Finally, I propose a Viterbi-style algorithm for considering both individual sentences and connection between sentences to create a coherence text.

I performed user studies to evaluate if the generated fabulas are coherent and if the natural language texts can express different narration styles, including the interestingness of the text, the degree that it resembles languages in fiction, and positive or negative sentiments. The user studies show that (1) the fabulas generated are mostly coherent, event matching human-written stories on some measures of coherence, and (2) the sentence selection heuristics and the sentiment decisions made based on SSWN strongly correlate with human intuition. These positive results demonstrate the high quality of the learn knowledge representation and the versatile storytelling capabilities of the SCHEHERAZADE system.

The field of computational story generation and storytelling has been associated with the field of computational creativity (cf. [42, 66, 100]). Boden [13] has proposed a classification of three types of creativity: combinational, exploratory, and transformational. If we consider the plot graph as defining a space for all legal event sequences, SCHEHERAZADE explores sequences in this space and may be considered to be exploratory creativity. On the other hand, the plot graphs combines stories from different exemplar stories. SCHEHERAZADE selects events from a plot graph, and describes these events by selecting from crowdsourced sentences. Therefore, SCHEHERAZADE may also be considered to possess combinational creativity.

The work in this chapter has made important contributions to generative Narrative Intelligence. To the best of my knowledge, this is the first Narrative Intelligence system that can generate all three tiers of narratives, including fabula, sjuzhet, and natural language text, from automatically learned knowledge. The user studies highlight the quality and variety of generated stories, suggesting that the learning algorithm for plot graphs is effective. The system provides a framework that integrates both plot-level and language-level generation of stories and provides flexible controls for the style of narration to be adjusted according to the needs of applications.

CHAPTER V

UNDERSTANDING STORIES

> *The ninety percent of human experience that does not fit into established narrative patterns falls into oblivion.*
>
> — Mason Cooley

In previous chapters, I demonstrate the utility of the learned knowledge representation (i.e. plot graphs) by implementing and evaluating techniques that generate and tell stories based on the learned knowledge. In this chapter, I further demonstrate that the Scheherazade system can understand stories based on plot graphs, which capture patterns of common situations. In particular, I focus on the problem of inferring the fabula from a given sjuzhet, defined as the capability U2 in narrative intelligence in Section 1.2. As discussed earlier, when a human tells a story, only a selected set of events that happened in the narrative world are told. The events being told constitute the sjuzhet. As an audience trying to fully understand the story, an AI system needs to infer the events that happened, which constitute the fabula.

I frame story understanding in terms of questions and answers. Answering questions has been a popular method for computational systems to demonstrate story understanding (c.f. [38, 73]). The Scheherazade system answers questions regarding events that have not been told to the system based on events that haven been told to the system. I first formally define the story understanding problem, and then prove its computational complexity to be NP-hard. After that, I present methods to reduce the computational complexity and solve the problem efficiently. The performance gains are evaluated against different sets of random graphs.

5.1 The Story Understanding Problem

As explained in Section 2.1, the story being told by human storytellers is a *sjuzhet*, not a *fabula*. Humans rarely tell every event that happened in the narrative world, but only those are needed for an interesting story. Thus, a common type of narrative understanding problem is to figure out what really happened in the narrative world based on what has been told to the AI. This ability to infer events that happened in the past and events that will happen in the future based on limited perception is important for making sense of the world. Consider the following example questions:

- Sally loved John. John asked Sally to marry him. Did Sally say yes?
- John covered his face and entered a bank. Later, John ran away from the bank. Did John demand money from the bank teller?
- John demanded money from the bank teller. The bank teller pressed the silent alarm. Did John escape from the bank?
- In a restaurant, John ordered food. After a while, John paid and left. Did John get food?

The above questions first posit that some events in the story domain have happened. The system needs to determine the probability of some other events happening, and should provide answers such as "the probability of John demanding money from the bank teller is 98%", or "It is very likely that John got food". It is worth emphasizing that these questions are about if an event happens in the fabula, not if the event is included in the sjuzhet. The ability to answer those questions demonstrate that the system possesses Narrative Intelligence about the domain.

Recall our definition for a plot graph $G = \langle E, P, M_x, E_o, E_c \rangle$, where E, T, M_x, E_o, E_c are the sets of events, precedence relations, mutual exclusion relations, optional events, and conditional events respectively. As explained in the previous chapter, a legal event sequence is constructed by walking the plot graph according to these simple rules:

1. An event e is eligible to be added to the sequence (or simply, event e "happens") if it has not parents or all of its parents meet one of the conditions: (1) the parent is optional (2) the parent has been deleted from the graph, or (3) the parent has been added to the sequence.

2. When an event e is added to the sequence, all other events that are mutually exclusive to e are removed from the plot graph.

3. Direct parents of removed events become direct parents of direct children of removed events.

4. If all parents of an event have been removed from the plot graph, that event is also removed and cannot be added to the sequence.

5. We stop adding events to the sequence when no events may be added, or when we reach one ending event in the plot graph.

Let $\mathbb{S}(G)$ denote all legal event sequences that the graph G can generate based on the above rules. Each sequence $S \in \mathbb{S}(G)$ is a sequence of events $\langle e_1, e_2, \ldots, e_k \rangle$ where $e_1, e_2, \ldots, e_k \in E$. When we have a set of events $F \subseteq E$, and the sequence S contains all events in F, we write $F \subseteq S$.

As explained earlier, in this chapter, I focus on story understanding as the inference of probability of events that are not mentioned in a story, given the knowledge that some other events in the situation have happened. I now define the Story Understanding Problem (SUP) formally:

Definition 12 (The Story Understanding Problem)**.** *Given a plot graph $G = \langle E, T, M_x, E_o, E_c \rangle$, one set of events $E_{required} \subseteq E$, and a query event e_{query}, the Story*

Understanding Problem determines the probability of e_{query} occurring in an event sequence s randomly drawn from $\mathbb{S}_G$ that includes all events in $E_{required}$.

The SUP is equivalent to finding the conditional probability $P(e_{query} \in S | E_{required} \subseteq S, S \in \mathbb{S}(G))$. For simplicity, we write it as $P(e_{query}|E_{required}, G)$.

Events in a plot graph can have complex interactions, and it is not straightforward to compute this probability directly from the graph (it will become clearer when we consider some of these interactions in Section 5.3). One method for solving this problem is to generate all possible legal event sequences, and count the number of sequences containing $E_{required}$ and e_{query}. Let $\mathbb{S}^r(G)$ denote the set of legal event sequences that contain $E_{required}$, and $\mathbb{S}^q(G)$ denote the set of legal event sequences that contain $E_{required} \cup \{e_{query}\}$. It is obvious that $\mathbb{S}^r(G)$ and $\mathbb{S}^q(G)$ are subsets of $\mathbb{S}(G)$. Assuming each event sequence is equally likely, we can compute the required probability as the ratio of the set cardinalities.

$$P(e_{query}|E_{required}, G) = \frac{\text{card}(\mathbb{S}^q(G))}{\text{card}(\mathbb{S}^r(G))} \tag{51}$$

where card$(\cdot)$ is the cardinality function. When there are no legal sequence containing all events in $E_{required}$, the conditional probability is undefined.

Equation 51 assumes each event sequence is equally likely in the real world, which admittedly may not always hold. Suppose we can obtain preferences over the events sequences as a weight function $w(S) \geq 0$ for every event sequence $S \in \mathbb{S}(G)$, such that $P(S) \propto w(S)$, the probability may be computed as

$$P(e_{query}|E_{required}, G) = \frac{\sum_{S \in \mathbb{S}^q(G)} w(S)}{\sum_{S' \in \mathbb{S}^r(G)} w(S')} \tag{52}$$

The weight function can take any form provided that $w(s) \geq 0, \forall s \in \mathbb{S}(G)$. Learning probabilities of event sequences directly from crowdsourced corpora may be difficult as the crowdsourced stories are sjuzhets rather than fabulas. This problem is out of the scope of the current dissertation.

As the next section shows, the SUP problem is NP-Hard, so it is unlikely that we will find a polynomial time algorithm for the SUP problem. Generating all possible sequences in $\mathbb{S}_G$ is expensive, as the number of all possible sequences is exponential to the number of all vertices in a graph in the worst case. However, it is still possible to improve the performance of story understanding over the brute force method by utilizing structures of the plot graphs. Such methods are developed in Section 5.3.

5.2 NP-Hardness of the Story Understanding Problem

In order to prove SUP is NP-hard, we first consider a simplified version of SUP.

Definition 13 (The Simplified Story Understanding Problem)**.** *Given a plot graph G containing n events and one set of events $E_{required} \subseteq E$, the Simplified Story Understanding Problem (SSUP) determines if there a legal event sequence that contains all events in $E_{required}$.*

Instead of finding a conditional probability (i.e. a ratio of two probabilities), the SSUP finds a true/false answer. When $E_{required} = \varnothing$, the SSUP is trivially true.

The Simplifed Story Understanding Problem (SSUP) is a weaker form of the SUP, since if we can determine in SUP that the probability of a legal event sequence containing some required events is defined and non-zero, we can conclude there must be at least one such event sequence for SSUP.

More specifically, we can perform a simple reduction from any SSUP to an SUP as follows: Given a SSUP with $E_{required} \neq \varnothing$, choose any event $e \in E_{required}$. Let $E'_{req} = E_{required} \setminus e$. The new SUP determines the probability of e occurring in legal event sequences containing all events in E'_{req}. The SSUP has an affirmative answer if and only if the answer to the constructed SUP is defined and non-zero.

Consequently, if SSUP is NP-complete, it follows that solving the Story Understanding Problem is NP-hard. Now we show the NP-completeness of the Simplified

Story Understanding Problem. The proof contains two steps: (1) SSUP is in NP and (2) SSUP is NP-hard.

NP is the set of computational problems that can be verified in polynomial time. When presented with any event sequence $S = \langle e_1, e_2, \ldots, e_k \rangle$, we can verify if S is legal and contains all events in $E_{required}$. We need to do the following:

1. Check if $k \leq n$. If $k > n$, the sequence s must not be legal. The time complexity is $O(1)$.

2. Check if $e_1, e_2, \ldots, e_k \in E$. This time complexity of this step, using a simple scanning algorithm, is $O(kn)$.

3. Check if the sequence contains all events in $E_{required}$, i.e. $E_{required} \subset s$. Similar to the previous step, the time complexity of this step is $O(kn)$.

4. We construct another legal event sequence S' by adding events in S one at a time, from the beginning to the end of S. When adding each event, we check if the event being added is eligible according to the story generation rules. It is obvious that each event can be checked in $O(n)$ time, and the total time complexity for this step is $O(n^2)$.

As $k \leq n$, the totally time complexity of checking a solution is $O(n^2)$. Therefore, the Simplified Story Understanding Problem is in NP.

We now show the 3-Satisfiability problem (3-SAT) can be reduced to the Simplified Story Understanding Problem in polynomial time. As 3-SAT is NP-complete, if any 3-SAT problem can be reduced to a SSUP in polynomial time, then SSUP must be at least as difficult as 3-SAT. Thus, SSUP is NP-hard. Since we have shown SSUP is in NP, it must also be NP-complete.

Definition 14 (3-Satisfiability)**.** *Consider a boolean formula which can be written as a number of conjunctive clauses, each containing exactly three boolean variables, such*

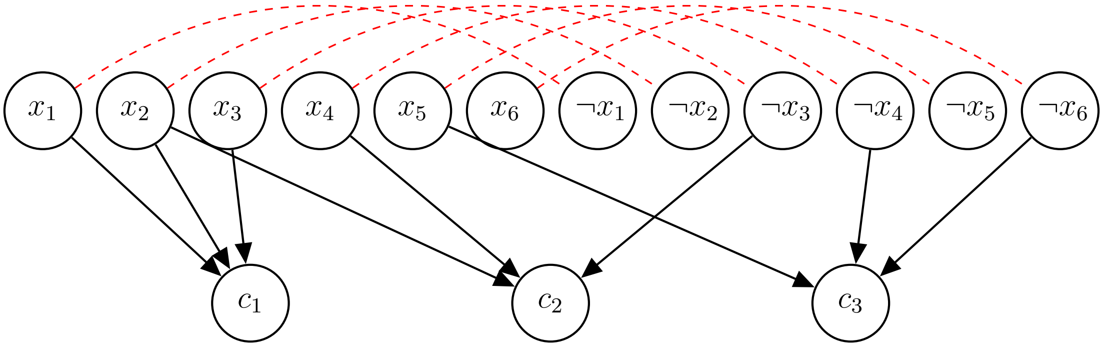


Figure 17: Reducing a 3-Satisfiability problem $f = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_5 \vee \neg x_6)$ to a plot graph.

as $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_5 \vee \neg x_6)$. *Is there an boolean assignment that makes the formula true?*

Given a 3-SAT boolean formula, we can construct the following plot graph: For each boolean variable in the formula, we create two vertices on the graph corresponding to the variable and its negate. Take the example of the formula $f = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_5 \vee \neg x_6)$, we would create 12 vertices for $x_1, \neg x_1, x_2, \neg x_2, x_3, \neg x_3, x_4, \neg x_4, x_5, \neg x_5, x_6$, and $\neg x_6$. For a formula involving n variables, we will have $2n$ vertices. We create a mutual exclusion relation between each variable vertex and its negation, e.g. between the vertices x_1 and $\neg x_1$. Subsequently, we create one vertex for one clause in the formula, and add a precedence relation from each variable vertex to every clause that variable appears in. In this example, we will create three clause vertices c_1, c_2 and c_3. Figure 17 shows the plot graph we constructed. For a boolean formula with n variables and m conjunctive clauses, we have $2n$ variable vertices, n mutual exclusion relations between the variable vertices, m clause vertices, and $3m$ precedence relations, so the construction has time complexity $O(n + m)$. The constructed SSUP problem asks if a legal event sequence can contain all the clause variables. In our example, $E_{required} = \{c_1, c_2, c_3\}$.

We now establish correspondence between the 3-SAT problem and the constructed

Simplified Story Understanding Problem: There is a valid variable assignment satisfying the boolean formula *if and only if* there is a legal event sequence in the plot graph that includes all the clause vertices. That is,

$$\text{The 3-SAT has a Yes answer} \Leftrightarrow \text{The constructed SSUP has a Yes answer}$$

First, we study the forward direction: for each valid variable assignment satisfying f, we can find a legal event sequence including all the clause vertices. For f to be true, each conjunctive clause must be true, and at least one of the three variables in each clause must be true. For each variable x_i that is true, we add the vertex corresponding to x_i to the event sequence. For each variable x_i that is false, we add the corresponding negated vertex $\neg x_i$ to the event sequence. Since the variable vertices do not have parents, and we never add both x_i and $\neg x_i$, the additions are always possible. After adding these variables, vertices corresponding to variables taking false values will be removed from the plot graph. For each clause vertex, at least one parent vertex will have been added to the sequence, so that clause vertex will be kept in the plot graph, rather than deleted recursively. Thus, we can add all clause vertices to the event sequence, and produce a legal event sequence satisfying the SSUP.

Next, we study the reverse direction: for a legal event sequence including all the clause vertices, we can find a valid variable assignment satisfying f. Since the event sequence includes all clause vertices, it must also include at least one of its three parents. Setting that variable to true will make that conjunctive clause to true. Thus, we can find a variable assignment that make each conjunctive clause true, and thereby satisfying the entire boolean formula.

Having established the equisatisfiability of 3-SAT and SSUP on the newly constructed plot graph, we conclude the SSUP is NP-complete and SUP is NP-hard.

5.3 Simplifying Plot Graphs

One way to solve the Story Understanding Problem is to generate all possible stories, or event sequences, allowed by the plot graph G, and count how many sequences contain all events in the set $E_{required}$ (i.e. the cardinality of $\mathbb{S}^r(G)$), and how many sequences contain all events in $E_{required}$ as well as the event e_{query} (i.e. the cardinality of $\mathbb{S}^r(G)$). The ratio of the two counts is the required probability $P(e_{query}|E_{required}, G)$. However, generating all stories in a plot graph is a very expensive operation. If the plot graph is large, sequences containing the set $E_{required}$ may be only a small fraction of all possible stories, so much of the computation is wasted. We would like to cut the number of stories we generate and reduce the overall computational cost. That is, we want to create a new plot graph $G_{simplified}$ from the original plot graph $G_{original}$ such that

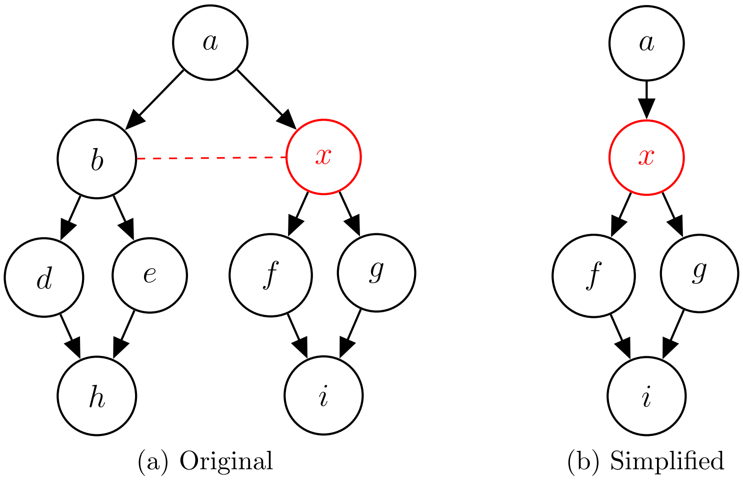
$$P(e_{query}|E_{required}, G_{simplified}) = P(e_{query}|E_{required}, G_{original}), \tag{53}$$

and

$$\text{card}(\mathbb{S}(G_{simplified})) < \text{card}(\mathbb{S}(G_{original})). \tag{54}$$

Assuming P $\neq$ NP, we will not be able to find a polynomial time algorithm for the Story Understanding Problem. However, we can still simplify the plot graph and reduce the amount of computation. If an event $a \in E_{required}$ is involved in a mutual exclusion relation with another event b, in many cases we can remove b from the plot graph as b will never happen in event sequences we care about. One such example is shown in Figure 18. In this particular case, $E_{required} = \{x\}$, so vertex b, being mutually exclusive to x, is removed from the plot graph. Children of b, including vertices d, e, and h, are also removed due to transitive closure. The simplified graph avoids the generation of event sequences *abdeh* and *abedh*, thereby saving computation.

When we remove unreachable vertices from the original plot graph $G_{original}$ to



(a) Original

(b) Simplified

Figure 18: An example of plot graph simplification due to mutual exclusion relations. $E_{required} = \{x\}$, as shown in red. We know vertex x must exist in the event sequence, so we can remove vertex b. Children of b are also removed due to transitive closure.

create the simplified plot graph $G_{simplified}$, we must make sure (1) $G_{simplified}$ can generate every event sequence that contain $E_{required}$ from $G_{original}$, or more formally,

$$\forall S \in \mathbb{S}(G_{original}), E_{required} \subseteq S \rightarrow S \in \mathbb{S}(G_{simplified}) \tag{55}$$

and (2) every event sequence that $G_{simplified}$ can generate is legal according to $G_{original}$, i.e.

$$\mathbb{S}(G_{simplified}) \subseteq \mathbb{S}(G_{original}) \tag{56}$$

These two conditions guarantee Equation 53 is satisfied. We do not require every event sequence in $\mathbb{S}(G_{simplified})$ to contain $E_{required}$.

Several considerations exist when we perform static analysis of plot graphs based on mutual exclusion relations. The first is what events can be removed given the execution of some other events. The second is how we regularize the event sequences in the graph where some events have been removed, so we do not create illegal event sequences. In both issues, we also need to consider race conditions that cannot be determined in static analysis. Some of these computations may take exponential time in the worse case, but their running times for typical plot graphs are usually short. In addition, the results of these computation can be stored together with the learned plot graphs, so story understanding can be performed quickly when called upon. Details of these computation will be discussed in the three subsections below.

5.3.1 Cause-for-Removals

The first task we need to do is to find out what events in the plot graph are capable of deleting other events, taking transitive closure into consideration. We define the notion of a Cause-for-Removal.

Definition 15 (Cause-for-Removal)**.** *Given a plot graph $G = \langle E, T, M_x, O \rangle$, a Cause-for-Removal (CfR) for a vertex $e \in E$ is a minimal set of vertices $E_c \subseteq E$ such that when all vertices in E_c are executed, e will be removed from the plot graph. As E_c is minimal, there is no vertex $d \in E$ such that the execution of events $E_c \setminus d$, i.e. all events in E_c except d, will remove d from the plot graph.*

This definition is straightforward. For example, if two vertices u and v are mutually exclusive to each other, u is a CfR for v and v is a CfR for u. If we denote the set of vertices mutually exclusive to u as $M_x(u)$, the condition is equivalent to $v \in M_x(u)$. Below are the scenarios that vertex u is a CfR for vertex v:

1. when u and v are directly involved in a mutual exclusion relation. That is, $v \in M_x(u)$

2. when u is mutually exclusive to *all* parents of v. If we denote the set of parent vertices of vertex v as $Pa(v)$, this condition can be written as $Pa(v) \subseteq M_x(u)$.

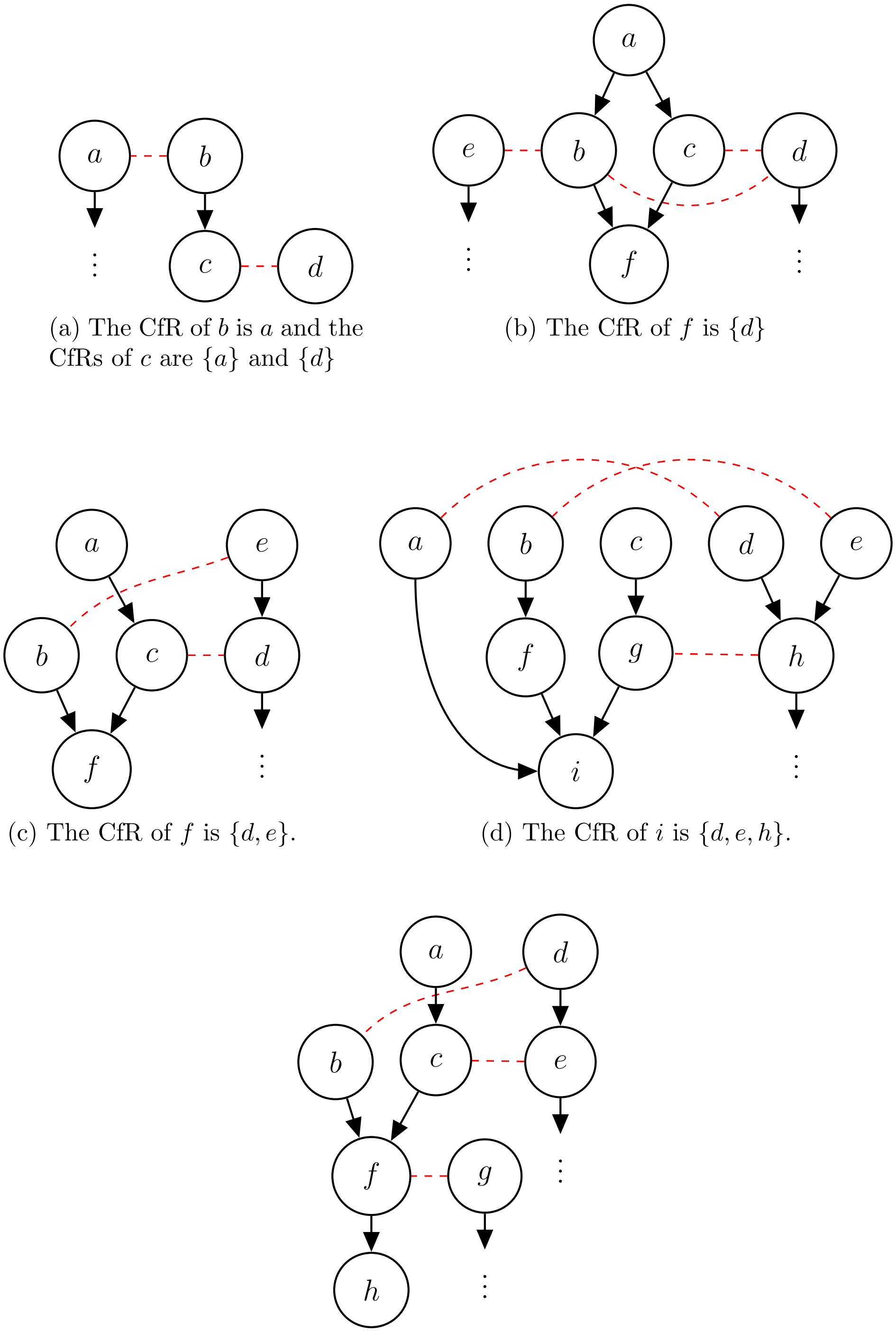
These two conditions seem obvious. Figure 19 shows some examples. In Figure 19(a), vertex a is a CfR for both vertices b and c, and vertex d is a CfR for vertex c. In Figure 19(b), we note vertex e is not a Cause-for-Removal of vertex f, because e cannot remove both parents of f at the same time. In fact, the existence of e creates a race condition. When e is executed before d, vertex b is deleted and a precedence from a to f is created. Due to this new precedence relation, the execution of d cannot remove all parents of f. I will discuss more about race conditions in Section 5.3.2.1.

However, a small modification, shown in Figure 19(c), produces a completely different result. Neither vertex e or d can delete all parents of vertex f, but we made two key changes: Vertex b does not have any parents, so deleting b will not create a new precedence relation. Vertex e is ordered before d, so that vertex c will be removed after vertex b. If c is removed before b, it will create a new precedence from a to f, which prevents f from being removed. Note that if e is not ordered before d, a race condition between e and d can occur.

Intuitively, in order to recursively remove a vertex v from the plot graph, we must remove all of its parents and prevent any new precedence relations from its grandparents. When vertex v has grandparents, we must cut any ties with its grandparents by executing a single event. Otherwise, new precedence relations will be created and the removal of v is prevented.

We first consider CfRs containing only a single vertex. Based on the above intuition, we can put these vertices into two categories. Category-A (Cat-A) vertices remove a vertex and all of its predecessors. Category-B (Cat-B) vertices remove a vertex but not all of its predecessors. In Figure 19(d), vertex d is a CfR for vertex a. Since a does not have any predecessors, d is a Category-A vertex for a. Vertex d removes f and its parent b, so d is a Cat-A vertex for f and b. Vertex h removes g but not its parent c, so vertex h is a Cat-B vertex for g. The reader may have noticed that the definition of Cat-A and Cat-B vertices depends on the vertices being removed.

The division between Cat-A and Cat-B vertices is the most important insight in detecting Cause-for-Removals. It is critical for combining multiple vertices into a single CfR, as discussed below, and for detecting race conditions, as discussed in Section 5.3.1.1.



(a) The CfR of b is a and the CfRs of c are $\{a\}$ and $\{d\}$

(b) The CfR of f is $\{d\}$

(c) The CfR of f is $\{d, e\}$.

(d) The CfR of i is $\{d, e, h\}$.

(e) The CfRs of h are $\{d, e\}$ and $\{g\}$.

Figure 19: Examples of causes for removal

When we combine single-vertex CfRs to create a multi-vertex CfR by removing all parents of a vertex, we need to note that in a multi-vertex CfR, there can be many Cat-A vertices but only one Cat-B vertex, and this Cat-B vertex must execute after all Cat-A vertices. Since Cat-A vertices remove a vertex and all of its predecessors, we do not worry about cutting anything from its predecessors. On the other hand, Cat-B vertices does cut a vertex off from its predecessors. So we must finish this cutting with one single vertex. To form a valid Cause-for-Removal, the Cat-B vertex must be preceded by all Cat-A vertices. If some Cat-A vertices are preceded by the Cat-B vertex, the vertices cannot form a CfR. If some Cat-A vertices are parallel to the Cat-B vertex, we have a race condition. I will discuss race conditions in Section 5.3.2.1.

We can now examine Figure 19(d) and 19(e). In Figure 19(d), when considering CfRs for vertex i, we see it does not have direct mutually exclusive vertices. Hence, the only way to remove i is to remove all of its parents. Parent vertices a and f can be removed by Cat-A vertices, and parent vertex g can only be removed by h, a Cat-B vertex. Since h is preceded by both d and e, we can take the union set $\{d, e, h\}$m which forms a valid CfR for vertex i. In Figure 19(e), vertex f can be removed by vertex g, which is a Cat-B vertex. In addition, its parent b can be removed by d, a Cat-A vertex. Another parent c can be removed by vertex e, a Cat-B vertex. As d precedes e, the union $\{d, e\}$ is another valid CfR for f. Vertex f is the only parent of vertex h, so CfRs of f are also CfRs of h.

5.3.1.1 Race Conditions

Race conditions happen during the simplification of plot graphs because although we know which events are required in the event sequences, we do not know their relative order. This lack of information may lead to some indeterminacy. A race condition in the detection of Cause-for-Removals can happen between a group of

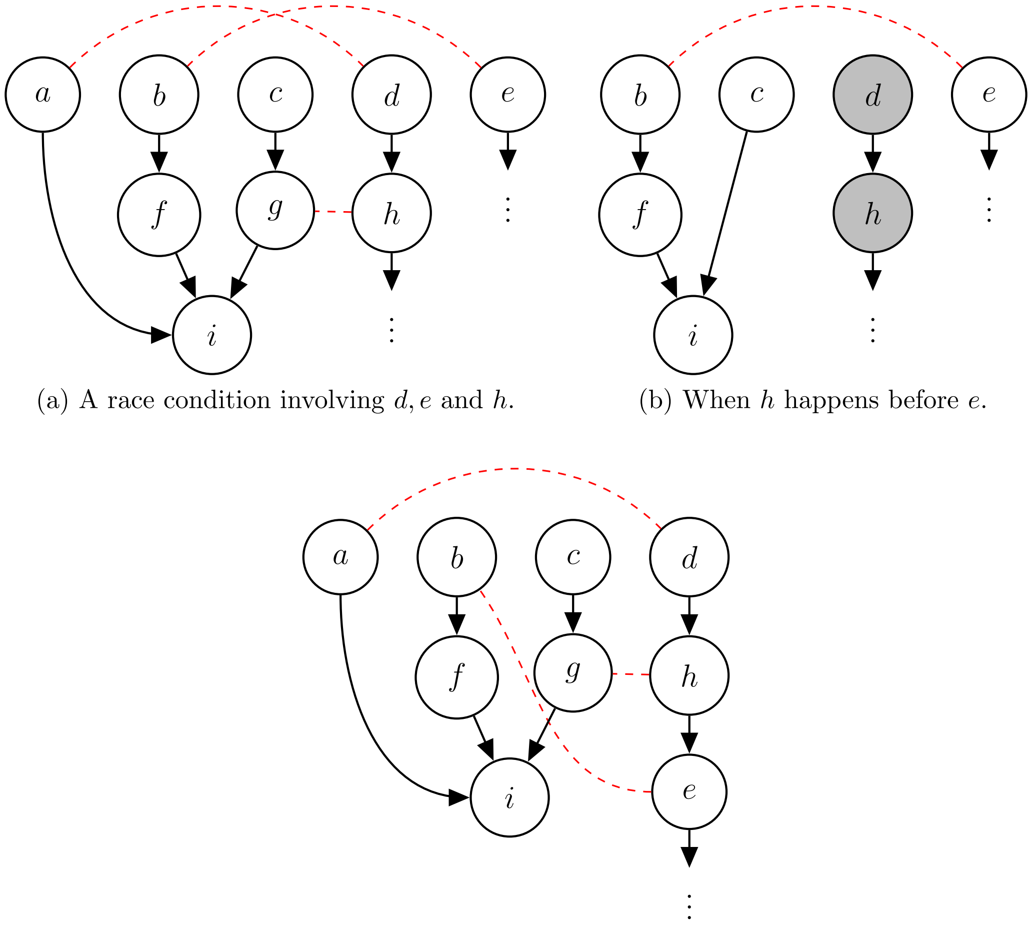
Cat-A vertices and a Cat-B vertex, as well as between two Cat-B vertices. There are no race conditions between Cat-A vertices because any two Cat-A vertices can work together in any order to remove other vertices.

We first examine the competition between Cat-A and Cat-B vertices. With some modification to Figure 19(d), we produce an example shown as Figure 20(a). Figure 20(a) removes the precedence relation from vertex e to vertex h in Figure 19(d). Thus, the order of execution between e and h is left unspecified. If we execute the vertices in the order $\langle deh \rangle$ or $\langle edh \rangle$, the vertex i will be removed as all of its parents have been removed. However, if we executes the events in the order $\langle dhe \rangle$, a precedence relation will be created between c and i immediately after executing h. The execution of vertex e will not remove vertex i from the graph. In summary, the ordering of execution can affect whether a vertex is removed or not. Figure 20(c) contains no race conditions. Since h always executes before e, vertex i will never be removed. There is neither a CfR nor a race condition.

Thus, a race condition can happen for a set of vertices U if the following conditions are satisifed:

- U contains some Cat-A vertices and one Cat-B vertex
- The order of execution is unspecified between some Cat-A vertices and the Cat-B vertex.
- If the Cat-B vertex is executed after all Cat-B vertices, some other vertices in the graph will be removed. That is, when properly ordered, U is a CfR for some vertices.

Race conditions can happen between Cat-B vertices. We have seen one such example in Figure 19(b), which is reproduced as Figure 21(a). Vertex d is a CfR for both vertices b and c, so it is CfR for their common child, f. Vertex e can only remove the vertex b but not c, so it is not a CfR for f. However, if e is executed before d,

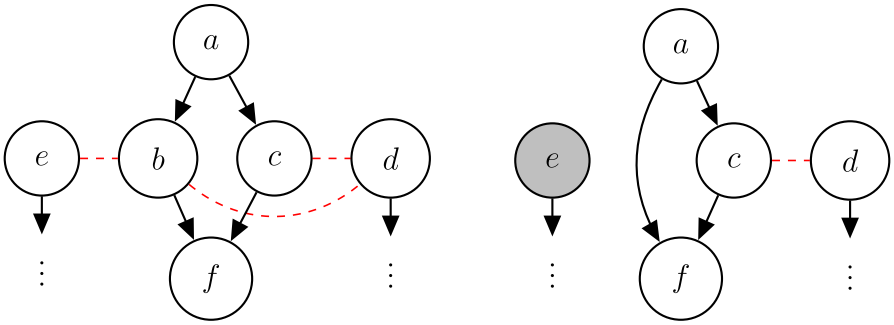


(a) A race condition involving d, e and h.

(b) When h happens before e.

(c) No race conditions nor CfRs.

Figure 20: A race condition during the detection of Cause-for-Removals, resulting in the indeterminacy whether a vertex can be removed. This race condition happens between Cat-A vertices and Cat-B vertices.Vertices that have been executed are shaded.



(a) Original

(b) e executes first. f will be kept when d subsequently executes

Figure 21: Another race condition during the detection of Cause-for-Removals, resulting in the indeterminacy whether a vertex can be removed. This race condition happens between two Cat-B vertices. Vertices that have been executed are shaded.

a new precedence relation will be created from a to f, preventing f to be removed later. Note vertex d is a Cat-B vertex for vertices b and c, and vertex e is a Cat-B vertex for vertex b.

Stating the above intuition formally, given a vertex v, and two Cat-B vertices a and b that remove parents of v, the two vertices a and b are in a race condition if

- Neither a or b are in direct mutual exclusion with vertex v.
- One of a and b removes some parents of v but not all parents of v. Without loss of generality, we assume a satisfies this condition.
- The other vertex, b, removes v by removing all parents of v.
- The order of a and b are unspecified. Suppose $a \prec b$, and a cannot be skipped in the graph (i.e. $a \notin E_o \cup E_c$), b is not a valid Cause-for-Removal.
- a and b are not involved in the same mutual exclusion relation.

When a race condition exists in a plot graph, the correct simplification will depend on the order of occurrence of competing vertices. The order of competing vertices can determine whether the children of the removed vertex can still happen, and when they can happen. Instead of creating two different plot graphs, I do not remove the vertex being contended in the race condition.

The algorithm for detecting CfR is sketched in Algorithm 7. The algorithm processes vertices in the order of a topological sort of the graph (function TopoSort), which guarantees we process parents before children. Clearly, CfRs of one vertex may affect the CfR of its children, but not the other way around. The algorithm first handles vertices without parents. These nodes cannot be removed by removing all of their parents, and every vertex mutually exclusive to them is necessarily a Cat-A vertex. If a vertex has some parents, we first try to find Cause-for-Removals that remove all of its parents. After that, vertices that are directly mutually exclusive are recorded as Cat-B CfRs.

The MergeParentsCfR attempts to find all CfRs that can remove the parents of a given vertex v. It starts with CfRs of one parent, and checks if this CfR is compatible with other CfRs that removes other parents. If the two CfRs are compatible, they are merged. If they form a race condition, the race condition is also noted. This function may potentially perform a combination of all CfRs of all parents, and the number of combinations is exponential to the number of parents in the worst case. However, most CfRs are not compatible and most vertices in realistic plot graphs do not have more than 4 or 5 parents. Thus, this algorithm terminates quickly in practice.

The conditions for two CfRs to be compatible are stated as follows: for two CfRs c_1 and c_2 that removes some parents of vertex v, they are compatible if all of the following are true.

- Either c_1 or c_2 contains no Cat-B vertices, or they contain the same Cat-B vertex.

- If a Cat-B vertex exists in either c_1 or c_2, it must be preceded by all other vertices in both c_1 and c_2.

- No two vertices in c_1 and c_2 are involved in the same mutual exclusion relation.

The conditions for two CfRs to be in a race condition are stated as follows: for two CfRs c_1 and c_2 that removes some parents of vertex v, they are in a race condition if all of the following are true.

- Either c_1 or c_2 contains a Cat-B vertex, or they contain the same Cat-B vertex.
- The Cat-B vertex is parallel to some other vertices in c_1 or c_2, but never precedes any vertices in c_1 or c_2.
- When vertices c_1 and c_2 are executed in the correct order, they can remove v.
- No two vertices in c_1 and c_2 are involved in the same mutual exclusion relation.

Another possible speed-up comes from the simplification of Cause-for-Removals by casting them to boolean formulae. Figure 22 shows two examples where the CfR for vertex g can be simplified. In Figure 22(a), the vertex f has two CfRs: $\{c\}$ and $\{d\}$. The vertex e has one CfR: $\{d\}$. When we combine the CfRs of the parents of vertex g, we obtain $\{d\}$ and $\{c, d\}$. It is worth noting that a list of CfRs can understood as a boolean formula, where the vertices in each CfR are conjunctive because all must be executed to remove a vertex, and different CfRs are disjunctive because any, if executed, may remove the vertex. With slight abuse of notation, we can write the above list of CfR as $(c \wedge d) \vee (d)$. We can easily see that $(c \wedge d) \vee (d) = d$. This simplification process reduces the number of CfRs that need to participate in the combination and accelerates computation. The simplification algorithm is shown in Algorithm 8.

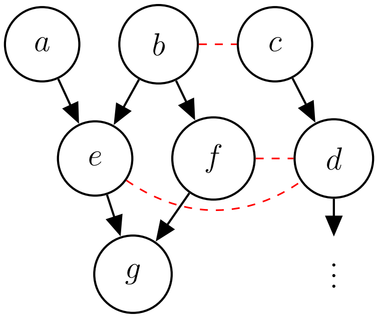
5.3.2 Implied Temporal Relations

The removal of a vertex from a plot graph removes a precondition for all its children that remain in the plot graph. Such removals may lead to the generation of event

Algorithm 7 Detecting Causes for Deletion

```
function DetectCfR($G = \langle E, P, M_x, E_o, E_c \rangle$)
    $order \leftarrow$ TopoSort($G$)
    create a dictionary $D$ that maps each vertex to a set of CauseForRemovals
    for each $v \in order$ do                                ▷ Processing in topological order
        if ParentsOf($v$) $= \varnothing$ then
            if $\neg v \in E_o$ then
                for each $m = (u, v) \in M_x$ do ▷ skip optional events with no parents
                    create a new CfR $c$, $c$.CatA $= \{u\}$
                    add $c$ to $D(v)$
                end for
            end if
        else
            $pd \leftarrow$ MergeParentsCfR($G$, $D$, $v$)                ▷ A set of CfRs
            add $pd$ to $D(v)$
            for each $m = (u, v) \in M_x$ do
                if $u$ has not been used in any CfR in $D(v)$ then
                    create a new CfR $c$, $c$.CatB $= u$
                    add $c$ to $D(v)$
                end if
            end for
        end if
    end for
    return $D$
end function

function MergeParentsCfR($G$, $D$, $v$)
    $parents \leftarrow$ ParentsOf($G, v$)
    $active \leftarrow D(parents.head)$
    while $active \neq \varnothing$ do
        $parents \leftarrow parents.tail$
        $next \leftarrow D(parents.head)$
        for each $c \in active$, each $d \in next$ do
            if Compatible($c, d$) then
                $e \leftarrow$ Merge($c, d$)
                $next\text{-}active \leftarrow next\text{-}active \cup \{e\}$
            else if RaceCondition($c, d$) then
                create a new race condition $rc$, $rc$.focus $\leftarrow v$, $rc$.foes $= \{c, d\}$
                add $rc$ to the global list of race conditions
            end if
        end for
        $active \leftarrow next\text{-}active$
    end while
    return $active$
end function
```



(a) The CfRs of g are $\{d\}$ and $\{c, d\}$, which is equivalent to {d}

Figure 22: Simplifying Cause-for-Removals

Algorithm 8 Simplifying CfR Boolean Formulas

```
function SIMPLIFY($CL = \{\{v^1_1, v^1_2, \ldots, v^1_k\}, \{v^2_1, v^2_2, \ldots, v^2_k\} \ldots\}$)
    for each clause $c_i = \{v^1_1, v^1_2, \ldots, v^1_k\} \in CL$ do
        Remove duplicate elements in clause
        if $\exists$ clause $c_j \in CL, i \neq j, c_j \subseteq c_i$ then
            $CL \leftarrow CL \setminus \{c_i\}$
        end if
    end for
    Remove duplicate clauses in $CL$
    return $CL$
end function
```

sequences that are not legal in the original plot graph. To solve this problem, we need to explicitly represent precedence relations that are implicit in the original plot graph.

Let us consider the situation shown in Figure 23(a). Suppose $E_{required} = \{x\}$, so we can remove vertex b from the plot graph. The child of b, vertex e, will not be removed from the plot graph as one of its parents, vertex c, still remains in the plot graph.

If we simply remove b from the plot graph, we obtain an incorrect simplification shown in Figure 23(c). In the original graph (Figure 23(a)), vertex e can only happen after either (1) that both parents b and c happen or (2) that c and x happen, since x removes b as a precondition for e. In the incorrect simplification in Figure 23(c), vertex e can happen before vertex x.

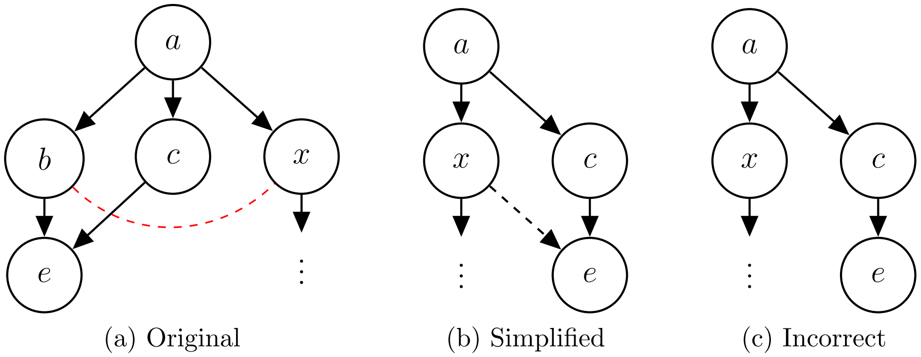


Figure 23: An example plot graph where a temporal relation must be added to ensure the correctness of the simplified graph. $E_{required} = \{x\}$, shown in red. When vertex x happens, vertex b will be removed, so vertex e will have one less precondition. To ensure that e only happens after x, we must add an explicit temporal relation from vertex x to vertex e. The naive simplification in (c) allows vertex e to happen before vertex x.

The problem is that the original graph contains an implied precedence relation from x to e because x removes one precondition for e. In general, an implied temporal relation exists between a vertex e, and a group of vertices $U = \{u_1, \ldots, u_k\}$ when all of the following conditions are satisfied:

- U is a Cause-for-Removal of at least one parent p of e, and p is not optional or conditional. Conditional or optional events are not preconditions for any vertices and can be removed without affecting its children.
- U is not a Cause-for-Removal of e. That is, $\neg CfR(U, e)$.
- All vertices in U are unordered w.r.t. e, i.e. $\forall u_i \in U, u_i \parallel e$

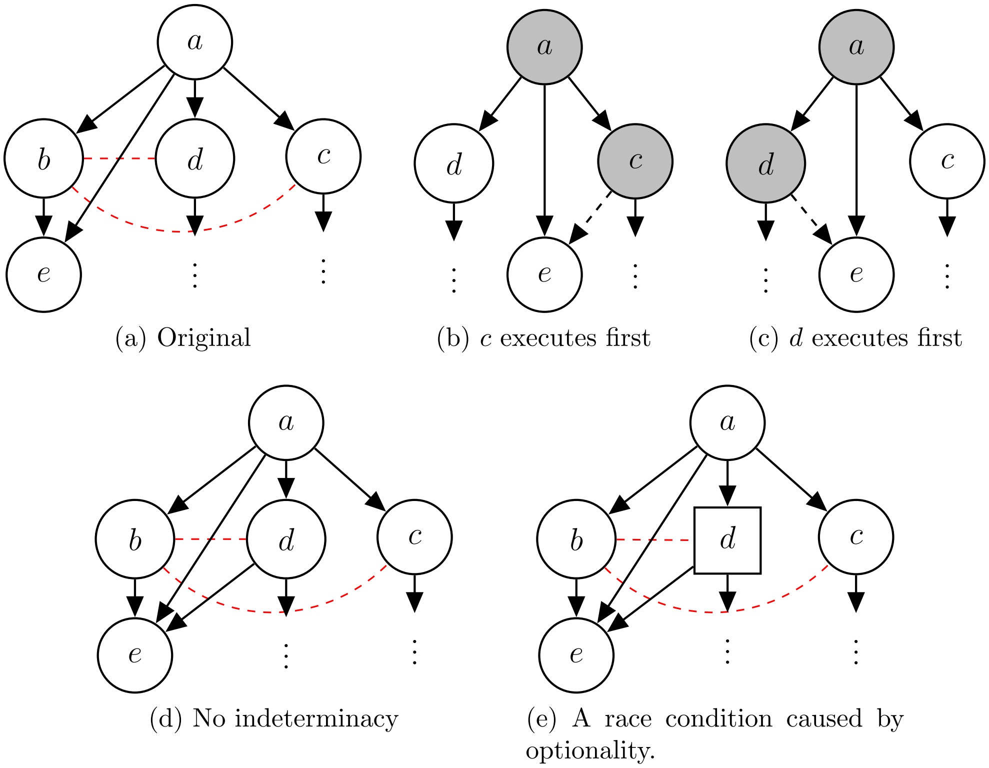
To represent this implied precedence relation, when U only contains one single vertex, we can explicitly add a precedence relation to the graph. This newly added directed edge is shown as the dashed arrow in Figure 23(b). This added edge would prevent the generation algorithm from generating an illegal sequence. It is important to note a difference between this newly added precedence relation and explicit precedence relations existing in the original plot graph: the newly added precedence relations cannot be used to determine transitive closure of deletion. If for some reasons we need to delete vertex c, we must also delete e due to transitive closure; we cannot treat x as a parent of e. To differentiate this newly added relation and the traditional precedence relations that can be used to determine transitive closure of deletion, we call this new type of relations *temporal relations*.

However, when U contains more than one vertex, we do not have a general method to represent this implied precedence relation on the graph. Hence, we perform a filtering step *after* generating all event sequences. This filtering removes all sequences where e executes before any vertices in U.

5.3.2.1 Race Conditions

Race conditions can also happen in the detection of implied temporal relations. Figure 24 shows an example where we cannot decide if and where we need to add a temporal relation. When either vertex c or vertex d executes, vertex b will be removed. Between vertex c or vertex d, it is the earlier vertex that removes a precondition for vertex e. As we discussed in Section 5.3.2, we must add a temporal relation to the graph, but we do not know which vertex, c or d, is the earlier vertex. If vertex c executes before vertex d, the new temporal relation should go from c to e, as shown in Figure 24(b). If vertex d executes before vertex c, the new temporal relation must go from d to e, yielding the simplified graph in Figure 24(c). If we do not add the temporal relation, we may generate the sequence $\langle aexy \rangle$ or $\langle aeyx \rangle$, which are not allowed in the original graph.

This indeterminacy of temporal relation ceases to exist when either c or d is ordered before e and is not optional. Without loss of generality, Figure 24(d) shows the case where vertex d is ordered before vertex e. In this case, c is free to happen before or after d. This is because when b does not happen, d is guaranteed to happen before e. Illegal sequences $\langle aecd \rangle$ and $\langle aedc \rangle$ are not possible. The indeterminacy also



(a) Original

(b) c executes first

(c) d executes first

(d) No indeterminacy

(e) A race condition caused by optionality.

Figure 24: An example of race condition during mutual exclusion analysis concerning the addition of temporal relations. Depending on the order of occurrence between vertex c and vertex d and if they are optional, we may have to add temporal relations in the simplified plot graphs. Vertices that have executed are shaded.

does not exists when either c or d is ordered after e. If e precedes c, for example, we will know that we must add a temporal relation from d to e, resolving the indeterminacy. If both vertices c and d are ordered w.r.t. e, it is evident that no temporal relations are needed and no race conditions exist.

However, the race condition comes back when the vertex (either c or d) preceding e is made optional, as shown in Figure 24(e). Vertex d is ordered before vertex e but is also optional. Since d is optional, it is not required to execute. We again have two correct but incompatible cases: When c executes before d, it will remove b, and we do not have to add any temporal relations. This may generate the sequence $\langle adec \rangle$. When we decide to skip d, b is kept in the graph and is removed by c, so a temporal relation from c to e is needed. This may generate the sequence $\langle ace \rangle$. The temporal relation is needed here to avoid the illegal sequence $\langle aec \rangle$. Therefore, the race condition reappears.

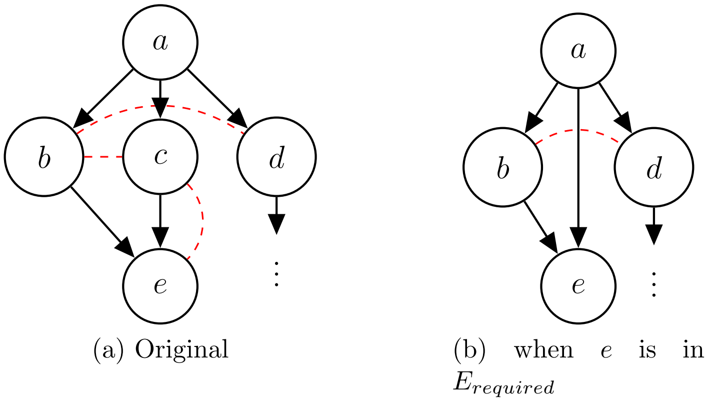
Stating the above intuition, the conditions for having a race condition during the detection of implied temporal relations are:

- Two CfRs, c_1 and c_2 can remove one parent p of vertex v, but neither c_1 and c_2 can remove v.

- The order of execution between c_1 and c_2 are not specified. That is, $\forall a \in c_1, \nexists\, b \in c_2, a \prec b$ and $\forall b \in c_1, \nexists\, a \in c_2, a \prec b$.

- The order of execution between c_1 and v are unspecified. That is, $\forall a \in c_1, \neg a \prec v \wedge \neg v \prec a$. The orders of execution between all vertices in c_2 and v are also unspecified. Without loss of generality, If the order between c_1 and v is specified, then all vertices in c_1 must be optional or conditional.

- No vertices in c_1 and c_2 are involved in a mutual exclusion relation.



(a) Original

(b) when e is in $E_{required}$

Figure 25: An example of of implied co-occurrence in the static analysis of mutual relations. In the original graph (a), vertex e can only execute when b executes. In the simplified graph (b), we can execute e without b.

5.3.3 Implied Preconditions

A third concern in the static analysis of mutual exclusion relations is implied preconditions that must also be made explicit after simplification. One example of this situation is shown in Figure 25(a). Note in the graph vertex c and e are ordered by a precedence relation and are mutually exclusive, but they are not identified as optional or conditional. This is because the existence of vertex b provides a path to vertex e, which gives e a chance to execute. In this plot graph, vertex e can only execute when vertex b executes. When we know vertex e must execute, we can remove vertex c from the plot graph, resulting in Figure 25(b). However, removing vertex c allows vertex e to execute without vertex b. For example, we can generate the sequence $\langle ade \rangle$ where d removes b.

The problem is that vertex b is an implied precondition in the original graph, which is lost during our simplification. However, we do not have a graph construct that would guarantee a precondition in plot graphs. To solve this issue, I perform an extra filtering step that discards all sequences where the implied preconditions are violated.

The conditions for the existence of an implied precondition between vertex a and

b are:

- Vertex c is ordered before vertex b, and b and c are mutually exclusive.
- Vertices c and b are not recognized as optional and conditional because of vertex a, which provides an alternative path to b.

5.3.4 Algorithm

In this section, I outline the overall algorithm for the static analysis of mutual exclusion relations (shown as Algorithm 9). The function SimplifyGraph takes five parameters: $G, C, R, IT, IP, E_{required}$ and E_{query}. $G = \langle E, P, M_x, E_o, E_c$ is the original plot graph. D is a dictionary that maps each vertex in E to a set of Cause-for-Removals. R is a list of race conditions. Each race condition contains two sets of vertices ($R.set1$ and $R.set2$) that compete for one vertex ($R.focus$). IT is a set of implied temporal relations. Each temporal relation contains a set of vertices E_{prior} and another vertex E_{later}, where vertices in E_{prior} is constrained to execute before E_{later}. IP is a set of implied preconditions.

The algorithm contains many bookkeeping steps and requires some explanation. The second line finds $E_{deferred}$. When a CfR contains a Cat-B vertex and one or more Cat-A vertices, the Cat-B vertex must execute later. Therefore, if $E_{required}$ does not contain all Cat-A vertices in the CfR, we must not remove vertices that can be removed by the Cat-B vertex alone. $E_{deferred}$ contains these Cat-B vertices.

Line 4 finds all vertices that may be removed from the plot graph, captured by $E_{possible}$. Line 5 finds $E_{remaining}$. If one vertex is being contended in a race condition, this vertex must not be removed from the plot graph. $E_{remaining}$ contains these vertices. Therefore, $E_{possible} \setminus E_{remaining}$ are the vertices to be removed from the plot graph. Line 8, 9, and 10 handle the insertion of implied temporal relations.

The Count function generates all possible vertex sequences from the simplified graph $G_{simplified}$. Note that a valid sequence must begin with a valid source vertex

Algorithm 9 The Algorithm for Mutual Exclusion Analysis

```
function SIMPLIFYGRAPH($G, D, R, IT, IP, E_{required}, E_{query}$)
    $all\text{-}cfr \leftarrow$ all CfRs in $D$
    $E_{deferred} \leftarrow \{cfr.\text{CatB} \in all\text{-}cfr \mid cfr.CatA \nsubseteq E_{required}\}$
    $E_{active} \leftarrow E_{required} \setminus E_{deferred}$
    $E_{possible} \leftarrow$ vertices that can be removed by $E_{active}$
    $E_{remaining} \leftarrow \{r.focus \mid r \in R \wedge (r.set1 \subseteq E_{required} \vee r.set2 \subseteq E_{required})\}$
    $E_{removed} \leftarrow E_{possible} \setminus E_{remaining}$
    $G_{simplified} \leftarrow$ REMOVEEVENTS($G, E_{removed}$)

    $T \leftarrow \{t \in IT \mid t.E_{prior} \subseteq E_{required}\}$
    $T_{added} \leftarrow \{(t.Eprior(0), t.E_{later}) \mid t \in T, |t.E_{prior}| = 1\}$
    $G_{simplified}.P \leftarrow G_{simplified}.P \cup T_{added}$

    $E_{sources} \leftarrow$ source nodes in G
    $E_{ends} \leftarrow$ end nodes in G
    return COUNT($G_{simplified}, E_{sources}, E_{ends}, IT, IP, E_{required}, E_{query}$)
end function

function COUNT($G_{simplified}, E_{sources}, E_{ends}, IT, IP, E_{required}, E_{query}$)
    $seqs \leftarrow$ all possible vertex sequences starting from $E_{sources}$ and ends in $E_{ends}$ or ends when no more vertices may be executed
    $valid\text{-}seqs \leftarrow$ vertex sequences where $IT$ and $IP$ constraints are respected
    $s_1 \leftarrow$ number of sequences in $valid\text{-}seqs$ containing $E_{required}$
    $s_2 \leftarrow$ number of sequences in $valid\text{-}seqs$ containing $E_{required}$ and $E_{query}$
    return $s_1/s_2$
end function
```

in the original graph G, and it must end with a valid end vertex in the original graph G, or when no more vertices may be executed. The generation algorithm has been shown as Algorithm 5. The generated sequences are further filtered with the implied temporal relations and implied precondition constraints. Finally, we count the sequences and compute the ratio.

5.4 Evaluation

I evaluate the effectiveness of the algorithm for plot graph simplification by testing them against random plot graphs with random precedence relations and mutual exclusion relations.

5.4.1 Methodology

The algorithm for generating random plot graphs takes three parameters: the total number of vertices, total number of precedence relations, and total number of mutual exclusion relations. As the plot graph is a directed acyclic graph, the vertices are first numbered from 1 to n. Precedence relations are uniformly drawn from all possible precedence relations, which go from a vertex with a smaller number to a vertex with a greater number. After the precedences are created, we omit precedences that are implied by transitive closure (e.g. $a \prec b$ and $b \prec c$ implies $a \prec c$.) Similarly, to generate the mutual exclusion relations, I uniformly drawn from all possible vertex pairs. Finally, optional and conditional vertices are detected. For each set of parameters, 10,000 random plot graphs are generated.

In order to test the effectiveness of the analysis based on mutual exclusion relations, from each random plot graph, I pick one vertex involved in at least one mutual exclusion relation as $E_{required}$. Another vertex is uniformly drawn from all vertices as E_{query}.

For each randomly generated plot graph, I record the total number of generated sequences and the amount of computation time for the original graph and the simplified graph respectively. For the original plot graph, all computation time is spent on generating event sequences. For the simplified plot graph, the computation time contains the time spent on graph simplification and generating the event sequences for the simplified plot graph. The averages over 10,000 graphs are computed as the sum from all simplified graphs divided by the sum from all original graphs.

5.4.2 Results

The acceleration obtained for random plot graphs generated by different sets of parameters are shown in Table 18. The sets of parameters were selected so that the random graphs resemble the actual plot graphs we have learned. As the number of

Table 18: Acceleration obtained by simplifying plot graphs based on mutual exclusion relations

Configuration			Savings		Error Rate
Vertices	Precedences	Mutual Exclusions	Sequences	Time	
10	20	3	29.47%	27.03%	1.56%
10	25	3	30.19%	23.78%	0.63%
10	30	4	30.34%	22.74%	0.66%
12	25	4	31.04%	32.97%	2.86%
12	35	4	31.31%	31.34%	1.05%
14	40	4	31.84%	32.93%	1.75%
14	50	6	32.32%	33.34%	2.09%
16	60	6	32.85%	36.29%	2.61%
16	70	6	32.62%	32.34%	1.52%
16	70	8	33.06%	34.03%	2.54%

vertices grows, the number of precedence relations and mutual exclusion relations are also increased to maintain a similar level of parallelism and mutually exclusive alternatives in the plot graphs. The configurations are generally sorted in increasing order of complexity.

The results indicate that the simplification of plot graphs based on mutual exclusion relations can achieve substantial increase in performance. On average, the knowledge of one event involved in a mutual exclusion relation can reduce the total number of sequences generated by more than 30%. The average saving on computation time grows from a low of 22.74% to a high of 36.78% when the number of vertices grows from 10 to 16. The error is greater than zero across all tests, indicating the algorithm does not always simplify plot graphs correctly. However, the error rate never exceeded 3%.

5.4.3 Discussion

One important purpose of the graph simplification algorithm is to effectively perform story understanding in large, complex plot graphs. As the number of legal vertex sequences grows superlinearly with the number of vertices, it is important to reduce the time of computation when the number of vertices becomes large.

Observing Table 18, we note the simplification algorithm scales well. As plot graphs become more complex, the saving in number of generated vertex sequences tends to increase, although the speed of increment is slow. The saving on the actual computation time portrays a more optimistic picture. The saving on time increases by 7% to 14% when the number of vertices increases from 10 to 16. This indicates the simplification algorithm only incurs a small computation overhead, and the overhead is dominated by the growth of vertices. Overall, this indicates the simplification algorithm can effectively reduce the computation needed to answer the Story Understanding problem.

It must be acknowledged that the current simplification algorithm does not simplify all plot graphs correctly. It is likely there are other forms of race conditions or implied temporal relations that I have not discovered. However, the error rate does not always increase as the plot graph grows more complex. Moreover, mistakes are only made on less than 3% of all plot graphs. In other words, given a random plot graph, we are at least 97% confident that the simplification algorithm will produce a correct answer. Other approximation algorithms, such as Monte Carlo methods that sample sequences from plot graphs, are left for future work.

5.5 Summary

In this chapter, I have investigated the Story Understanding Problem and its mathematical properties. Noting that the sjuzhet of a story often hides details in the fabula from the audience, I examined the inference of unmentioned events based on explicitly mentioned events in the story, by utilizing the plot graphs SCHEHERAZADE has learned earlier.

I provided a mathematical definition for the Story Understanding Problem and further showed that it is NP-hard. However, by utilizing properties of the plot graph representation, it is possible to reduce the needed computation time by simplifying the plot graph, if we know some events that exist in the story are also involved in some mutual exclusion relations. Employing this information, we can remove vertices from the plot graph, thereby reducing the amount of computation spent on generating all possible event sequences. I have discussed and solved many issues in performing this type of static analysis of plot graphs, including finding Cause-for-Removals, race conditions, implied temporal relations and preconditions. Experimental results suggest the simplification algorithm scales well when the plot graph grows more complex. On average, simplifying plot graphs can reduce the total number of event sequences by 29.47% to 33.06%, and the total computation time by 27.03% to 36.29%.

In conclusion, the simplification algorithm demonstrates good scalability that allows us to efficiently answer the Story Understand Problem. The ability to perform story understanding based on automatically learned knowledge holds the promise of scaling computational Narrative Intelligence to solving diverse and large real-world problems.

CHAPTER VI

FUTURE WORK AND CONCLUSIONS

> *If I've vividly laid out the narrative, the reader will come to his own conclusions.*
>
> — Rick Atkinson

In this concluding chapter, I summarize this dissertation and highlight its contributions. I then discuss two major unsolved problems that lie ahead on the road to strong computational Narrative Intelligence, and speculate about possible solutions.

6.1 Summary

Despite their differences, most existing computational Narrative Intelligence systems rely on hand-crafted knowledge, which requires considerable amount of expert labor. As a result, these systems are confined to a few domains where knowledge has been provided. This knowledge bottleneck has been widely recognized, but its solution has not been thoroughly investigated.

In this dissertation, I propose procedures and techniques for learning knowledge to support narrative intelligence, including story generation, storytelling and story understanding, in domains previously unknown to the system. The Scheherazade system is capable of extending its own knowledge when it encounters unfamiliar storytelling domains. I collect a number of simple exemplar stories in a given sociocultural or procedural situation (e.g. a date at a movie theater, pumping gas into a car, etc.) from workers on Amazon Mechanical Turk. The crowdsourcing approach allows us to circumvent natural language challenges and directly tap the collective social norms from the minds of members of a community.

A situational model, called a plot graph, can be learned robustly from the crowdsourced stories. A plot graph describes how a situation typically unfolds in terms of events, precedence relations, mutual exclusion relations, and optional/conditional events. The plot graph representation has proven to be versatile in supporting tasks of narrative intelligence. I have created a complete algorithmic pipeline for generating the fabula (i.e. all events that happen in the narrative world), the sjuzhet (i.e. the events actually being told) and the media (i.e. the final story in text). I also developed algorithms for efficiently inferring if an untold event has happened, given the events being told to the AI. Evaluations demonstrate that the system can (1) learn plot graphs accurately (Section 3.4 and 3.9), (2) generate stories that match human-written stories in some measures of coherence, marking an important milestone for computational Narrative Intelligence (Section 4.2), (3) tell the stories in distinctive narration styles (Section 4.5), and (4) reduce the computation time by approximately 30% or more in story understanding (Section 5.4). Consequently, the system can demonstrate narrative intelligence in any situation where a small number of exemplar stories may be collected.

6.2 Contributions

In this dissertation, I made the following contributions:

- The system addresses the knowledge bottleneck that has troubled Narrative Intelligence systems for decades. By learning from crowdsourced stories, the system is able to demonstrate Narrative Intelligence in any situation where a small number of stories can be collected. This is the first computational system that employ learned knowledge to perform both story generation and story understanding.

- I provide a novel knowledge representation, called a plot graph. This representation can be accurately learned using statistical methods and supports many

Narrative Intelligence tasks, thereby effectively bridging learning and application. This representation incorporates natural language processing into an NI framework.

- In the evaluation of fabula coherence, I demonstrate for the first time that computer-generated stories can match human-written stories in some aspects of coherence. This is also the first time that a computational NI system is directly compared to humans.
- The Scheherazade system learns diverse narration styles from large data sets, which enables applications in storytelling virtual characters.

6.3 Potential Applications

In addition to long-term scientific contributions, the development of the Scheherazade system provides immediate benefits in supporting several potential applications, including virtual training environments and virtual characters.

The Scheherazade system was initially inspired by the need to develop an intelligent and interactive training application for preparing children with autism for navigating everyday social situations, as autistic children often experience difficulties in understanding and following social conventions. During the development, it became apparent that the manual authoring approach will not scale up to complexities of real-world scenarios. Therefore, it is necessary for a computational system learning the knowledge by itself. Inspired by Boujarwah *et al.* [14]'s initial work on crowdsourcing scripts for the training environment, in this dissertation, I develop an entire procedural pipeline from collecting exemplar stories to the automatic learning of the knowledge and its applications.

The Scheherazade system's capability to learn socio-cultural conventions is not limited to the the mainstream culture where autistic children need to cope with, but can also extend to foreign cultures. For example, training scenarios can be developed to prepare users for job interviews in Japan or respecting local customs in Samoa. It could be expensive to hire experts on foreign cultures to manually author sufficient knowledge in support of virtual training environments. If the culture is not well studied, finding an expert on the subject could be a difficult task by itself. In comparison, SCHEHERAZADE only requires a small number of exemplar stories written by English-speaking non-experts who have some experience with the foreign culture and that particular social situation. No training in computer science or AI is required.

Moreover, the SCHEHERAZADE system enables the creation of diverse virtual characters in games or virtual worlds that can tell stories. For entertainment purposes, oftentimes we would like to create an illusion that virtual characters lead their own lives and are not just part of a show that disappears when the player looks away. Thus, we would like to give each character a background story or an alibi, which can explain where they have been and what they have done while they are not with the player [176]. Virtual characters should be able to tell these background stories and recall details to substantiate their stories when asked to. Therefore, the knowledge of common social situations, the ability to identify prototypical events, and the ability to tell stories in diverse narration styles can help the creation of believable virtual characters.

Virtual characters with background stories also have applications in the healthcare industry. For example, Bickmore *et al.* [12] found that the ability to tell autobiographical stories increases the likelihood that human users will interact with virtual characters over an extended period. This can be advantageous when we need to encourage users to keep interacting with some programs or electronic devices, such as educational software or medical devices for self-monitoring.

6.4 Future Work

The limitations of each individual components of the system have been discussed throughout this dissertation. In this section, I will discuss two major future directions: how to understand and generate creative stories, and how to further increase the flexibility of the knowledge representation.

6.4.1 Creativity

It can be argued that the stories currently generated by SCHEHERAZADE are not qualitatively creative, although the stories generated by the system can be different from any input story. As the plot graphs capture typical behaviors in typical situations, the stories generated by following the plot graphs are prototypical and lack surprises. Therefore, an important future direction is to extend the creative capabilities of SCHEHERAZADE.

It is possible to specifically crowdsource exemplar stories that break the norm. In an earlier, unsuccessful attempt to crowdsource stories for the movie date situation, for example, one creative story introduced a third character, a friend of John who insisted in sitting with the couple and disrupted their date. In later attempts, the crowdsourcing instructions were changed to be explicit that we only wanted stories about the most typical or mundane stories in a situation. However, the statistical nature of learning makes it difficult to learn from those creative stories. Among the stories that are very dissimilar from other stores, some are coherent and interesting, whereas others are actually nonsensical. From a pure statistical point of view, it is difficult to separate those two cases.

As a consequence, a major challenge for learning from creative exemplar stories is to understand creative stories. One possibility is to crowdsource the understanding. The crowd workers may be asked to vote for stories that are both coherent and creative, or repair nonsensical stories using a mechanism similar to the story repairing experiment in Section 4.2.

AI systems have attempted to understand creative stories. Ram [146] pointed out that most, if not all, creative story variations are derived by combining and modifying typical behaviors and scripts. My survey of the manga *Doraemon* provides evidence the creative process of most science-fictional gadgets is based on modifying and blending ordinary objects we encounter in our life [98, 101]. This process is similar to conceptual blending [189, 15], a cognitive paradigm for human creativity. Therefore, given enough plot graphs, it is plausible that a computational system can make sense of a creative exemplar story by combining different plot graphs. For example, if the system possesses a plot graph about a third wheel situation, it may understand the aforementioned creative story as a conceptual blend of the third wheel graph and the movie date graph. Those that the system cannot make sense of will be regarded as nonsensical. The ISSAC system [120] understands creative story based on a large amount of manually authored knowledge. It is an promising research direction to build such a system based on purely learned knowledge.

6.4.2 Multiplicity of Plot Graph Levels

Being able to move between different levels of abstraction is a strength of human cognition and provides benefits to Narrative Intelligence. A high-level representation of a situation provides a concise and intelligible summary. A low-level representation contains many details, such as movements of body parts and facial muscles, moment-to-moment thought processes, and so on. A skillful story writer use different levels of abstraction to zoom in the important details and fast-forward unimportant events, creating variations in the tempo of the narrative. These techniques are known as compression and expansion (See Section 2.1.1). One vivid example of the expansion technique is the so-called "bullet time". First showcased in the movie *The Matrix*, bullet time shows bullet dodging actions in hyper slow motion in order to achieve special artistic effects [143]. A high-level representation can capture commonalities between many seemingly different scenarios, such as representing that the gas pumping situation, the pharmacy situation, and the movie date situation all involve the purchase of merchandise. A high-level understanding can facilitate the technique of compression.

The current plot graph representation contains basically two levels: the events and their natural language descriptions. In Section 4.3, I provided the EventRank algorithm that identifies the most and the least prototypical events in the plot graph. A higher-level representation, which sits on top of the event level, can thus be partially simulated by extracting most prototypical events. However, the EventRank algorithm does not identify a hierarchical relationship between the higher level and the event level, which potentially harms the quality of story summaries it produces. For example, in the bank robbery situation, it is currently impossible to create a higher level "demand money" event that encompasses both the event "John demanded money" and the event "John handed Sally a note". This high-level event should be assigned a higher weight than any of those two events and be recognized as a very prototypical event.

Future work is also required to learn a lower level of representation that describes body movements. With this level, the system will be able to create detailed descriptions such as "John set his left foot into the bank's lobby". Understanding the body movements associated with each action may also allow us to reason about unintended consequences, such as John stepping on banana peels and falling.

6.5 Conclusions

Narrative Intelligence is a vital component of human intelligence. Computational replication of Narrative intelligence has important implications for the development of human-level AI and numerous commercial applications. Decades of research made it clear that computational NI systems that aim to scale beyond a few simple domains must be able to learn the needed knowledge by itself. In recent years, we have witnessed the emergence of several Open Narrative Intelligence systems [177, 166, 114]. The Scheherazade system presented in this dissertation is the first system that can utilize the learned knowledge in both tasks of story generation and story understanding, and the first system that can generate stories that approximate the coherence of human-written stories. Its capabilities have been tested and proved in several user studies and evaluations. As such, I conclude the thesis statement, first proposed in Section 1.4 have been achieved. The Scheherazade system marks an important milestone for scaling computational Narrative Intelligence to meet the challenges of the real world. Its development helped us understand important problems and identify possible solutions in the scientific pursuit of an Artificial Intelligence capable of crafting, telling, understanding, and responding appropriately to narratives.

APPENDIX A

SMOOTH SENTIWORDNET

This Appendiex shows the 100 most positive and 100 most negative words in Smooth SentiWordNet. The negative words are shown on the left column and the positive words are shown on the right column.

Lemma / POS	Sentiment	Lemma / POS	Sentiment
bone-chilling/JJ	-8.00399	silky/JJ	0.90291
shriek/NN	-6.73801	kiss/VB	0.90774
scop/VB	-6.7339	fulfil/VB	0.90887
wail/VB	-6.30141	information/NN	0.91023
panic/VB	-5.67925	purchase/VB	0.91234
sob/NN	-5.64176	arrange/VB	0.91776
scream/NN	-5.31574	cooperate/VB	0.92015
nightmare/NN	-4.95367	fully/RB	0.92036
terror/NN	-4.26091	wine/NN	0.92326
panic/NN	-4.04759	better/JJ	0.93538
weep/VB	-3.84039	satisfy/VB	0.9364
fright/NN	-3.7388	view/NN	0.93922
sob/VB	-3.56282	comfortable/JJ	0.93955
terrify/VB	-3.51428	topic/NN	0.93957
anger/NN	-3.4937	new/JJ	0.94163
victim/NN	-3.28723	friend/NN	0.94739
scream/VB	-3.28343	smile/VB	0.94977
screech/VB	-3.22727	pretty/JJ	0.95263
sweaty/JJ	-3.22524	dame/NN	0.96067
hysterically/RB	-3.01847	politics/NN	0.9745

flee/VB	-2.83545	interact/VB	0.97931
hate/VB	-2.80631	sport/NN	0.98264
bark/VB	-2.72129	lady/NN	0.98368
collapse/VB	-2.70325	present/VB	0.99649
dread/VB	-2.63414	important/JJ	1
yell/VB	-2.62926	menu/NN	1.00148
restroom/NN	-2.61856	stoplight/NN	1.00301
frightened/JJ	-2.56544	young/JJ	1.00905
stumble/VB	-2.52733	favorite/JJ	1.01332
freeze/VB	-2.51679	particularly/RB	1.01674
fear/NN	-2.503	yes/NN	1.01886
thirst/NN	-2.47606	best/RB	1.02546
thunder/NN	-2.42492	best/JJ	1.02959
death/NN	-2.23877	good/JJ	1.03473
dismay/NN	-2.17959	expensive/JJ	1.03554
clutch/VB	-2.09228	interest/NN	1.04276
violently/RB	-2.07236	blond/JJ	1.05009
struggle/VB	-2.02189	special/JJ	1.0502
scared/JJ	-2.00743	comfy/JJ	1.05107
scurry/VB	-1.95179	like/VB	1.06617
shocking/JJ	-1.93507	restoration/NN	1.0752
darkness/NN	-1.89049	bright/JJ	1.09547
sorrowful/JJ	-1.83753	favor/NN	1.09927
cause/VB	-1.68308	baseball/NN	1.10351
worst/JJ	-1.66493	glory/NN	1.10587
alarm/NN	-1.61991	appropriate/JJ	1.123
suspenseful/JJ	-1.61467	nameplate/NN	1.12736
burst/NN	-1.60499	credit/NN	1.13414
rush/VB	-1.58249	ecstatic/JJ	1.13983
heat/NN	-1.55584	love/NN	1.14522

ordeal/NN	-1.55112	mirror/VB	1.14923
frighten/VB	-1.42775	satisfactory/JJ	1.14993
convict/VB	-1.41959	music/NN	1.15872
darken/VB	-1.41634	greet/VB	1.16503
twisted/JJ	-1.35634	queue/VB	1.16565
wind/NN	-1.33197	sweet/JJ	1.17272
nervous/JJ	-1.32785	mutual/JJ	1.19994
tremble/VB	-1.31654	character/NN	1.2009
rush/NN	-1.30267	perfectly/RB	1.21689
sound/NN	-1.26086	chat/VB	1.22213
sunglass/NN	-1.22531	beverage/NN	1.22738
throat/NN	-1.21807	thorough/JJ	1.22821
tear/NN	-1.16682	impress/VB	1.23884
mortally/RB	-1.14252	delicious/JJ	1.2429
slam/VB	-1.13843	remarkable/JJ	1.24377
pound/VB	-1.12395	tasty/JJ	1.2681
dash/VB	-1.09977	initiate/VB	1.27167
llama/NN	-1.07874	coyly/RB	1.29285
darkened/JJ	-1.06664	epic/NN	1.3139
pretzel/NN	-1.05842	interesting/JJ	1.31427
sad/JJ	-1.05651	happy/JJ	1.34045
excruciating/JJ	-1.03506	perfect/JJ	1.34832
ice-cold/JJ	-1.02378	facility/NN	1.35465
rip/VB	-1.01851	rehearsed/JJ	1.35902
whip/VB	-1.0169	harmony/NN	1.39367
audible/JJ	-1	goer/NN	1.39632
robbery/NN	-0.98525	enthral/VB	1.40056
coarse/JJ	-0.95885	wonderful/JJ	1.41497
pack/NN	-0.95334	applaud/VB	1.46886
prison/NN	-0.94258	enjoy/VB	1.47755

distract/VB	-0.91207	romantic/JJ	1.51518
tense/JJ	-0.88605	blonde/JJ	1.51591
fall/VB	-0.88226	classic/JJ	1.52007
level/VB	-0.86619	presentation/NN	1.52359
escape/VB	-0.86512	clunker/NN	1.62548
leg/NN	-0.85497	flirtatious/JJ	1.66228
unburied/JJ	-0.85305	richly/RB	1.6989
rob/VB	-0.83481	delightful/JJ	1.78992
body/NN	-0.82224	preview/NN	1.82059
devour/VB	-0.79887	enjoyable/JJ	1.82074
loud/JJ	-0.79637	deal/NN	1.97416
irremediable/JJ	-0.77608	beam/VB	1.9844
ignition/NN	-0.77341	admire/VB	2.00495
heart-rending/JJ	-0.74582	immensely/RB	2.02764
fetid/JJ	-0.74403	sexy/JJ	2.13925
shoot/VB	-0.72868	captivate/VB	2.39255
nylon/NN	-0.7126	small-framed/JJ	2.99211
mangled/JJ	-0.70885	nacho/NN	3.14946
grievously/RB	-0.70578	snack/VB	3.2766
dank/JJ	-0.68328	tryed/JJ	4.62135

APPENDIX B

GENERATED STORY TEXTS

This section shows the stories generated for the purpose of the user study discussed in Section 4.5.2.

B.1 Stories in the Bank Robbery Situation

B.1.1 The Positive Story

John drove his grandmothers borrowed old clunker to the bank on Main St.

John took a deep breath and opened the bank door, letting an elderly woman exit before he entered himself.

John did not want to be recognized.

John entered the Yes bank at 1 pm.

John gave a thorough look around the bank to see how many people were inside.

John spotted a young blond teller, Sally, behind the counter.

John stood behind the lady and toddler and politely waited his turn, noticing the name plate on the counter... " Sally'.

John walked carefully up to the counter and interacted with Sally.

Sally started feeling the hairs on the back of her neck stand up.

John pulled out the gun, still smiling.

John pointed his gun at Sally to push the point of the seriousness.

Upon seeing the gun, Sally yelled out in fear.

John wanted more money.

John presented Sally with a bag in which to put the money.

Sally continued to cooperate, putting the money into the bag as ordered.

John pocketed the money.

Sally cried, somewhat relieved it may be over soon.

Sally described John as best as she could to the police.

John ran out of the bank and got in his grandmothers car.

John drove away, sure the police were behind him all the way back to his grandmothers house in the country.

B.1.2 The Negative Story

John drove to the bank, with a nervous look on his face.

John opened the bank door while his heart was beating fast.

John put on sunglasses.

John walked into the bank with a handgun underneath his jacket.

John looked around the bank, scoping out security cameras or guards.

John noticed one of the tellers named Sally seemed bored and distracted.

John stood in line.

John approached Sally naturally as to not raise alarm.

Sally saw Obama standing in front of her and she felt her whole body tense up as her worst nightmare seemed to be coming true.

John pulled out a gun.

John leveled the gun at Sally and kept it on her.

Sally let out a bone-chilling scream.

John barked his orders at Sally, demanding she put the money in the bag.

John forced the bag into Sallys hands.

Sallys hands were trembling as she put the money in the bag.

John then grabbed the bag of money out of Sallys nervous hands.

Sally felt tears streaming down her face as she let out sorrowful sobs.

Still shaken, Sally reached for the phone and in a panicked manner called the police.

John quickly fled the bank and entered into his car.

John escaped in the car.

B.1.3 The Concise Story

John drove to the bank.

John opened the bank door.

John did not want to be recognized.

John went into the bank.

John looked around the bank.

John recognized Sally.

John stood in line.

John walked to Sally.

Sally was scared.

John took out a gun.

John pointed gun at Sally.

Sally screamed.

John demanded $ 1,000,000 from Sally.

John gave a bag to Sally.

Sally put money into the bag.

John took the money.

Sally cried.

Sally called the police.

John got in his car.

John drove away in the car.

B.1.4 The Story with Most Interesting Details

John got into his car with his disguise, gun and note in his knapsack and headed towards the Old Second in the next town over, repeating his rehearsed demands silently over and over in his head.

John watched while a little old lady left the bank and walked to her car and then slipped on his gloves, slipped his gun into his coat pocket, grabbed his mask and strode determinedly to the lobby door and pulled it open.

John looked at his reflection in the glass of the door, gave himself a little smirk and covered his face.

John took another deep breath as he wondered if this was really a good idea, and entered the bank.

John looked around the bank, making sure his timing was right.

John spotted a young blond teller, Sally, behind the counter.

John stood behind the lady and toddler and politely waited his turn, noticing the nameplate on the counter... "Sally".

When it was his turn, John, wearing his Obama mask, approached the counter.

Sally saw Obama standing in front of her and she felt her whole body tense up as her worst nightmare seemed to be coming true.

Once Sally began to run, John pulled out the gun and directed it at the bank guard.

John wore a stern stare as he pointed the gun at Sally.

Sally screamed hysterically which alerted other people in the bank.

John demanded Sally to give her all of the money she had in her drawer, and all the money that was close that she could get to quickly.

John tossed the bag he had brought for the money at Sally.

Sally put the money in the bag, and collected the money from the 2 tellers next to her.

John struggled to stuff the money in his satchel.

Sally was quietly sobbing as John grabbed the bag full of money.

Sally called the cops.

John strode quickly from the bank and got into his car tossing the money bag on the seat beside him.

The car drove away from the bank.

B.1.5 The Story with Most Interesting Details and Length Penalty

John calmly drove to the bank to avoid drawing attention to himself.

John opened the door.

John covered his face.

John walked into the front door of the bank.

John looked around, scanning the bank for anything nearby.

John spotted a young blond teller, Sandy, behind the counter.

John stood behind the lady and toddler and politely waited his turn, noticing the nameplate on the counter... “Sally”.

Quietly and calmly John walked up to Sally’s window.

Sally saw the smile and got scared.

John reached behind his back and withdrew his pistol.

John wore a stern stare as he pointed the gun at Sally.

This caused Sally to let out an audible shriek.

John asked for all the money in the drawer.

John then handed Sally an empty sack.

Sally stuffed the money into John’s bag.

John pocketed the money.

Sally continued to sob hysterically as she sat down on her stool.

Sally called the cops.

John got into his red pickup truck and slammed the door.

John sped away, hoping to get distance between him and the cops.

B.2 Stories in the Movie Date Situation

B.2.1 The Positive Story

John drove his red car to Sally's house to pick her up for their movie date.

Sally, in her sexy yellow dress climbed into the sleek sports car.

John and Sally bought their movie tickets at " The Dame", a local theater that had been restored to its former 1940s glory.

John and Sally showed their tickets to the attendant who smiled and waited for them at the entrance to the theater.

Sally bought herself a small soda, while John got nachos and some sour patch kids.

John and Sally entered the theater and admired the big screen.

John and Sally bought delicious drinks.

John and Sally scoured the theater for the best seats in the house, and found a perfect pair right in the center of the theater.

John and Sally sat down in the velvet covered seats, and took in the richly appointed theater.

John and Sally chatted during the previews.

After the lengthy and irritating previews, the movie finally began.

Sally watched the opening scenes intently while she snacked on her popcorn.

John and Sally took a drink of their beverages to wash down the popcorn.

During the romantic parts of the movie, John pulled the classic "yawn and put arms around" move.

John and Sally sat captivated by the movie.

During the movie, the drinks really went through John and he needed to excuse himself.

John and Sally felt endlessly fulfilled by the movie and enjoyed it immensely.

John gave Sally a soft hug during the movie, and Sally returned the favor.

John and Sally felt romantic holding hands throughout the movie.

The movie had a delightful and surprising ending.

John and Sally stood up from their seats during the ending credits.

John and Sally walked through the theater and then through the parking lot to get to their car.

John and Sally left the theater and enjoyed the fresh night air.

John kissed Sally good night.

B.2.2 The Negative Story

With sweaty palms and heart racing, John drove to Sallys house for their first date.

Sally was glad to get out of the heat and get into the air conditioned car.

John and Sally arrived at the theater just before the movie was scheduled to start and rushed to buy their tickets for the movie.

John and Sally quickly whipped out their tickets for the ticket checker.

John and Sally decided that they wanted some popcorn.

John and Sally entered the darkened theater and paused letting their eyes adjust to the dimness.

John realized they had not bought drinks for themselves, which caused them both to laugh before they went back to the snack bar.

Stumbling through the darkness of the theater, John and Sally eventually found their seats.

John awkwardly grabbed Sallys coat and then they both sat down.

John made nervous small talk to Sally before the movie started.

The movie began with a loud sound of thunder.

The couple devoured the popcorn they ordered.

Because Johns throat became dry from the popcorn, John drank his soda.

John tried to put his arms around Sally, but she dodged him.

John and Sally watched the film.

John and Sally walked to the restrooms.

John and Sally sat with big smiles on their faces as they watched the movie.

After a sad moment in the movie, John reached out and hugged Sally.

Sally held Johns sweaty hands throughout the movie.

Much to Johns dismay, the movie ended.

Sally stood up first, followed by John, and they both stretched quickly to revive their bodies after a long time of sitting.

John and Sally slowly walked to the car and got inside, dreading the end of the date.

John drove the car out of the movie theater parking lot and turned toward Sallys house.

At the front door of Sallys house, John leaned in and gave Sally a peck on the lips.

B.2.3 The Concise Story

John drove his car to Sallys house.

Sally got into Johns car.

John got the tickets.

John and Sally gave the tickets to the usher.

Sally paid for the snacks.

Sally and John went into the theater.

John bought drinks.

Sally found seats.

John and Sally sat.

John and Sally talked about the movie.

The movie began.

John shared his popcorn with Sally.

Sally took a drink.

John puts his arm around Sally.

John and Sally watched the film.

John went to the bathroom.

John and Sally enjoyed the movie.

John hugged Sally during the movie.

John and Sally held hands during the movie.

The movie ended.

Sally got up.

John and Sally ran to their car.

John and Sally went outside the theater.

Sally kissed John.

B.2.4 The Story with Most Interesting Details

John drove his red car to Sally's house to pick her up for their movie date.

Sally carefully entered the car with a smile.

John and Sally arrived at the theater just before the movie was scheduled to start and rushed to buy their tickets for the movie.

The tattooed, male attendant gave John a smile and a wink as he whispered the words " what a catch," while John showed the tickets at the entrance.

John bought a large tub of popcorn and two boxes of malted milk balls, Sally's favorite.

John and Sally entered the darkened theater and paused letting their eyes adjust to the dimness.

John decided that he wanted to buy a beer, so he asked Sally if she wanted a glass of red or white wine and John proceeded to buy them.

John and Sally found seats in the back.

It was still a few minutes before show time so the theater was still empty, making it easier for John and Sally to move down the row to their seats.

Having sat down, John asked Sally about her day and they enjoyed a bit of small talk while waiting for the show to begin.

Just as Sally had finished relating the events of that day to John, the theater fully darkened, the screen was unveiled, and the show began.

John and Sally shared popcorn during the movie, feeding each other and laughing.

John and Sally drank the ice-cold sodas, which went perfectly with the buttered popcorn.

Finally working up the courage to do so, John extended his arm to embrace Sally. He was relieved and ecstatic to feel her move closer to him in response.

Both feeling unsure of the others' next move, they watched the movie in silence.

Sally stood up to use the restroom during the movie, smiling coyly at John before that exit.

John and Sally sat with big smiles on their faces as they watched the movie.

John hugged Sally during the movie, and she returned the hug.

Sally held John's sweaty hands throughout the movie.

The movie ended and John and Sally remained seated for a few minutes, waiting for the crowd to leave the theater.

Sally stood up first, followed by John, and they both stretched quickly to revive their bodies after a long time of sitting.

Still holding hands, John walked Sally back to his car through the maze of people all scurrying out of the theater.

John and Sally carefully left the movie theater parking lot, not wanting to be hit by another car.

John let go of Sally's hand and opened the passenger side door of his car for her but instead of entering the car, she stepped forward, embraced him, and gave him a large kiss.

B.2.5 The Story with Most Interesting Details and Length Penalty

John drove to Sally's house, and nervously checked himself out in the rearview mirror before ringing her doorbell.

Sally, waiting for John on the front porch, slid into the passenger seat with a smile.

John paid for Sally at the ticket booth.

John and Sally showed their tickets to the attendant who smiled and waited for them at the entrance to the theater.

John and Sally were surprised how expensive the snacks were.

John and Sally entered the darkened theater and paused letting their eyes adjust to the dimness.

Drinks in hand, John and Sally found two seats near the back row.

John and Sally sat in chairs.

John and Sally spoke curtly to each other about politics.

Once the movie began, John and Sally sat quietly.

John thought the popcorn was extra buttery.

John and Sally drank the sugary sodas.

John nervously put his arm around Sally and she snuggled into him.

John and Sally watched the movie silently.

Sally stood up to use the restroom during the movie, smiling coyly at John before that exit.

John and Sally laughed and smiled at the movie plot.

John hugged Sally during the movie, and she returned the hug.

John and Sally also held hands throughout the movie, even though John's hands were sweaty.

John and Sally smiled as the movie ended and the lights came on.

John and Sally slowly got up from their seats.

John and Sally walked to the car in the parking garage.

John held the door open for Sally as they exited the theater.

John and Sally very softly kissed.

REFERENCES

[1] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J., "OPTICS: Ordering points to identify the clustering structure," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 49–60, 1999. 3.3, 3.3.2, 4, 3.3.2

[2] Applebee, A., *The Child's Concept of Story: Ages 2 to 17.* University of Chicago Press, 1978. 1.3

[3] Bae, B.-C., Cheong, Y.-G., and Young, R. M., "Automated story generation with multiple internal focalization," in *Proceedings of the 2011 IEEE Conference on Computational Intelligence and Games*, pp. 211–218, 2011. 2.2.2

[4] Bae, B.-C. and Young, R. M., "A use of flashback and foreshadowing for surprise arousal in narrative using a plan-based approach," in *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling* (Spierling, U. and Szilas, N., eds.), pp. 156–167, 2008. 1.2, 2.1.1, 2.2.2

[5] Bal, M., *Narratology: Introduction to the Theory of Narrative.* Toronto, Canada: University of Toronto Press, 1997. 1.1, 2.1.1, 4

[6] Bamman, D., O'Connor, B., and Smith, N. A., "Learning latent personas of film characters," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Sofia, Bulgaria), pp. 352–361, Association for Computational Linguistics, August 2013. 2.2.4

[7] Bamman, D., Underwood, T., and Smith, N. A., "A bayesian mixed effects model of literary character," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Baltimore, Maryland), pp. 370–379, Association for Computational Linguistics, June 2014. 2.2.4

[8] Barthes, R., "Structural analysis of narratives," in *Image, Music, Text*, New York: Hill and Wang, 1977. 4.3

[9] Barzilay, R. and Lapata, M., "Modeling local coherence: An entity-based approach," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, (Stroudsburg, PA, USA), pp. 141–148, Association for Computational Linguistics, 2005. 2.5, 4.6

[10] Bengio, Y., Louradour, J., Collobert, R., and Weston, J., "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009. 2.4.1

[11] Berliner, T. and Cohen, D. J., "The illusion of continuity: Active perception and the classical editing system," *Journal of Film and Video*, vol. 63, no. 1, pp. 44–63, 2011. 2.1

[12] Bickmore, B., Schulman, D., and Yin, L., "Engagement vs. deceit: Virtual humans with human autobiographies," in *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, 2009. 6.3

[13] Boden, M., *The Creative Mind: Myths and Mechanisms.* Weidenfeld/Abacus & Basic Books, 1990. 4.7

[14] Boujarwah, F. A., Kim, J. G., Riedl, M. O., Arriaga, R. I., and Abowd, G. D., "Building a knowledge base to support the authoring of social skills instructional modules," in *Proceedings of the 2001 International Meeting for Autism Research*, (San Diego, CA), 2011. 1.4, 6.3

[15] Brandt, L. and Brandt, P. A., "Making sense of a blend," *Apparatur*, vol. 4, pp. 62–71, 2002. 6.4.1

[16] Branigan, E., *Narrative Comprehension and Film.* Routledge, 1992. 1.3, 1

[17] Brenner, M., "Creating dynamic story plots with continual multiagent planning," in *Proceedings of the 24th National Conference on Artificial Intelligence*, 2011. 2.2.1

[18] Brin, S. and Page, L., "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, pp. 107–117, 1998. 4.3.1

[19] Brody, S., "Clustering clauses for high-level relation detection: an information-theoretic approach," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2007. 2.4.1

[20] Bruner, J., *Acts of Meaning.* Cambridge, MA: Harvard University Press. 1, 1.2

[21] Bruner, J., "The narrative construction of reality," *Critical Inquiry*, vol. 18, pp. 1–21, 1991. 1.2

[22] Callaway, C. B. and Lester., J. C., "Narrative prose generation," *Artificial Intelligence*, vol. 139, pp. 213–252, 2002. 2.2.3

[23] Carlson, L., Marcu, D., and Okurowski, M. E., "Building a discourse-tagged corpus in the framework of rhetorical structure theory," in *Proceedings of the 2nd SIGDIAL Workshop on Discourse and Dialogue*, pp. 1–10, 2003. 2.4

[24] Cavazza, M., Charles, F., and Mead, S. J., "Generating content for scenario-based serious-games using crowdsourcing," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, 2002. 2.2.1

[25] Cerini, S., Compagnoni, V., Demontis, A., Formentelli, M., and Gandini, C., "Micro-WNOp: A gold standard for the evaluation of automatically compiled lexical resources for opinion mining," in *Language resources and linguistic theory: Typology, Second Language Acquisition, English Linguistics*, Milan, Italy: Franco Angeli Editore, 2007. 2.3

[26] Chambers, N. and Jurafsky, D., "Unsupervised learning of narrative event chains," in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 789–797, 2008. 2.4.1, 3.1, 3.6

[27] Chambers, N. and Jurafsky, D., "Unsupervised learning of narrative schemas and their participants," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL '09, (Singapore), pp. 602–610, Association for Computational Linguistics, 2009. 2.4.1, 3.6

[28] Chambers, N. and Jurafsky, D., "Template-based information extraction without the templates," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon), pp. 976–986, Association for Computational Linguistics, 2011. 2.4.1

[29] Chatman, S. B., *Story and Discourse: Narrative Structure in Fiction and Film*. Ithaca and London: Cornell University Press, 1978. 4.3, 4.3.3

[30] Chen, S., Nelson, M., and Mateas, M., "Evaluating the authorial leverage of drama management," in *Proceedings of the 5th AAAI Conference on AI and Interactive Digital Entertainment*, AAAI Press, 2009. 4.1.2

[31] Cheng, P., "From covariation to causation: A causal power theory," *Psychological Review*, vol. 104, pp. 367–405, 1997. 3.10

[32] Cheng, W., Rademaker, M., Baets, B. D., and Hüllermeier, E., "Predicting partial orders: Ranking with abstention," in *Machine Learning and Knowledge Discovery in Databases* (Balcázar, J., Bonchi, F., Gionis, A., and Sebag, M., eds.), pp. 215–230, Springer, 2010. 3.6.2

[33] Cheong, Y.-G., *A Computational Model of Narrative Generation for Suspense*. Ph.D. dissertation, North Carolina State University, Department of Computer Science, 2007. 1.2, 2.1.1

[34] Cheong, Y.-G. and Young, R. M., "Suspenser: A story generation system for suspense," *IEEE Transactions on Computational Intelligence and AI in Games*, 2014. 2.2.2, 2.2.4, 4.3

[35] Colby, B. N., "A partial grammar of eskimo folktales," *American Anthropologist*, no. 75, pp. 645–662, 1973. 2.2.1

[36] Coppersmith, D. and Winograd, S., "Matrix multiplication via arithmetic progressions," *Journal of Symbolic Computation*, vol. 9, no. 3, pp. 251–280, 1990. 3.8

[37] Cullingford, R. E., "SAM," in *Inside Computer Understanding: Five Programs Plus Miniatures* (Schank, R. C. and Riesbeck, C. K., eds.). 1, 2.2.4

[38] Cullingford, R. E., *Script Application: Computer Understanding of Newspaper Stories.* Ph.D. dissertation, Yale University, Department of Computer Science, 1978. Technical Report 116. 1.3, 2.2.4, 5

[39] Curtis, J., Baxter, D., Wagner, P., Cabral, J., Schneider, D., and Witbrock, M. J., "Methods of rule acquisition in the textlearner system," in *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pp. 22–28, AAAI. 2.2.4

[40] Dautenhahn, K., "The narrative intelligence hypothesis: In search of the transactional format of narratives in humans and other social animals," in *Cognitive Technology: Instruments of Mind* (Beynon, M., Nehaniv, C. L., and Dautenhahn, K., eds.), vol. 2117 of *Lecture Notes in Computer Science*, pp. 248–266, Springer Berlin Heidelberg, 2001. 1, 1.2

[41] Davis, M. and Travers, M., "A brief overview of the narrative intelligence reading group," in *Narrative Intelligence* (Mateas, M. and Sengers, P., eds.), pp. 27–38, John Benjamins, 2003. 2

[42] Davis, N., Li, B., O'Neill, B., Riedl, M., and Nitsche, M., "Distributed creative cognition in digital filmmaking," in *Proceedings of the 8th ACM conference on Creativity and Cognition*, pp. 207–216, ACM, 2011. 4.7

[43] de Albornoz, J. C., Plaza, L., and Gervás, P., "SentiSense: An easily scalable concept-based affective lexicon for sentiment analysis," in *Proceedings of the 8th International Conference on Language Resources and Evaluation*, LREC '12, 2012. 2.3

[44] de Marneffe, M.-C. and Manning, C. D., "Stanford typed dependencies manual," 2008. Revised 2013. Last retrived from nlp.stanford.edu/software/dependencies_manual.pdf. 3.3.1

[45] Dunbar, R., *Grooming, Gossip and the Evolution of Language.* Harvard University Press, 1997. 1.2

[46] Dundes, A., *The Morphology of North American Indian Folktales.* Folklore Fellows Communications, 1964. 2.2.1

[47] Dyer, M. G., "The role of affect in narratives," *Cognitive Science*, vol. 7, pp. 211–242, 1983. [U3]

[48] Elman, J. L., “Learning and development in neural networks: The importance of starting small,” *Cognition*, vol. 48, pp. 781–799, 1993. 2.4.1

[49] Elsner, M., “Character-based kernels for novelistic plot structure,” in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL ’12, (Stroudsburg, PA, USA), pp. 634–644, Association for Computational Linguistics, 2012. 2.2.4

[50] Elson, D., *Modeling Narrative Discourse.* Ph.D. dissertation, Columbian University, Department of Computer Science, 2012. 3, 2.2.3, 2.5

[51] Esuli, A., Baccianella, S., and Sebastiani, F., “SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining,” in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)* (Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., and Tapias, D., eds.), (Valletta, Malta), pp. 19–21, European Language Resources Association (ELRA), 2010. 2.3, 4.4.2

[52] Etzioni, O., Hanks, S., Daniel Weld, D. D., Lesh, N., and Williamson, M., “An approach to planning with incomplete information,” in *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pp. 115–125, 1992. 1.4.1

[53] Fairclough, C. and Cunningham, P., “A multiplayer case based story engine,” Tech. Rep. TCD-CS-2003-43, Department of Computer Science, Trinity College Dublin, 2003. 2.2.1

[54] Falkenhainer, B., K., F., and Gentner, D., “The structure-mapping engine: Algorithm and examples,” *Artificial Intelligence*, vol. 20, no. 41, p. 163, 1989. 2.2.1

[55] Finlayson, M., *Learning Narrative Structure from Annotated Folktales.* Ph.D. dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2011. 2.5

[56] Fivush, R., Kuebli, J., and Clubb, P. A., “The structure of events and event representations: A developmental analysis,” *Child Development*, vol. 92, pp. 188–201, 1992. 1.4.1

[57] Flanagan, O., *Consciousness Reconsidered.* Bradford, 1993. 1, 1.2

[58] Flower, L. and Hayes, J., “A cognitive process theory of writing,” *College Composition and Communication*, vol. 32, no. 4, pp. 365–387, 1981. 1.2

[59] Forster, E. M., *Aspects of the Novel.* Edward Arnold, 1927. 2.1, 2.1.1

[60] Francis, W. N. and Kučera, H., *Frequency Analysis of English Usage.* Houghton Mifflin, 1982. 3.3.1

[61] Friedemann, K., *Die Rolle des Erzählers in der Epik.* 1910. Reprint: Darmstadt 1965. 2.1

[62] Fujiki, T., Nanba, H., and Okumura, M., "Automatic acquisition of script knowledge from a text collection," in *Proceedings of the Tenth Conference on European chapter of the Association for Computational Linguistics*, EACL '03, (Budapest, Hungary), pp. 91–94, Association for Computational Linguistics, 2003. 2.4.1, 2

[63] García Landa, J. A., *Structural Narratology: An Introduction.* Longman, 1994. 2.1.1

[64] Genette, G., *Nouveau discours du récit.* Seuil, 1983. 2.1, 2.1.1

[65] Gerrig, R. J. and Bernardo, A. B., "Readers as problem-solvers in the experience of suspense," *Poetics*, vol. 22, pp. 459–472, 1994. 2.2.2, 2.2.4

[66] Gervás, P., "Computational approaches to storytelling and creativity," *AI Magazine*, vol. 30, no. 3, pp. 49–62, 2009. 4.7

[67] Gervás, P., Díaz-Agudo, B., Peinado, F., and Hervás, R., "Story plot generation based on CBR," *Journal of Knowledge-Based Systems*, vol. 18, pp. 235–242, 2005. 1.3, 2.2.1

[68] Gervás, P. and León, C., "Reading and writing as a creative cycle: The need for a computational model," in *Proceedings of the 5th International Conference on Computational Creativity*, (Ljubljana, Slovenia), 2014. 1.2

[69] Girju, R., "Automatic detection of causal relations for question answering," in *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question AnsweringMachine Learning and Beyond*, 2003. 2.4.1

[70] Gockley, R., Bruce, A., Forlizzi, J., Michalowski, M., Mundell, A., Rosenthal, S., Sellner, B., Simmons, R., Snipes, K., Schultz, A. C., and Wang, J., "Designing robots for long-term social interaction," in *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2199–2204, 2005. 1.2

[71] Gordon, A., Bejan, C., and Sagae, K., "Commonsense causal reasoning using millions of personal stories," in *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, (San Francisco, CA), AAAI Press, 2011. 2.4.1

[72] Graesser, A., Singer, M., and Trabasso, T., "Constructing inferences during narrative text comprehension," *Psychological Review*, vol. 101, pp. 371–395, 1994. 1.4.1, 4.3

[73] Graesser, A. C., Lang, K. L., and Roberts, R. M., "Question answering in the context of stories," *Journal of Experimental Psychology:General*, vol. 120, pp. 254–277, 1991. 5

[74] Grisham, T., "Metaphor, poetry, storytelling and cross-cultural leadership," *Management Decision*, vol. 44, pp. 486–503, 2006. 1.2

[75] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2014. 3.6.2, 3.6.3

[76] Haveliwala, T. H., "Topic-sensitive PageRank," in *Proceedings of the 11th International Conference on World Wide Web*, pp. 517–526, 2002. 4.3, 4.3.1

[77] Hedlund, J., Antonakis, J., and Sternberg, R., "Tacit knowledge and practical intelligence: understanding the lessons of experience," Tech. Rep. ARI Research Note 2003-04, United States Army Research Institute for the Behavioral and Social Sciences, Fort Belvoir, VA, 2002. 3.2

[78] Heilmann, J., Miller, J., Nockerts, A., and Dunaway, C., "Properties of the narrative scoring scheme using narrative retells in young school-age children," *American Journal of Speech-Language Pathology*, vol. 19, no. 2, pp. 154–166, 2010. 4.3

[79] Hilton, D. J. and Slugoski, B. R., "Knowledge-based causal attribution: The abnormal conditions focus model," *Psychological Review*, vol. 93, no. 1, pp. 75–88, 1986. 3.10

[80] Huang, J., "Maximum likelihood estimation of dirichlet distribution parameters," 2005. 3.3.3

[81] Hudson, J. A. and Shapiro, L. A., "From knowing to telling: The development of children's scripts, stories, and personal narratives," in *Developing narrative structure*, pp. 89–136, Hillsdale, NJ: Lawrence Erlbaum, 1991. 1.3

[82] Ipeirotis, P. G., Provost, F., and Wang, J., "Quality management on Amazon Mechanical Turk," in *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pp. 64–67, 2010. 3.2

[83] Johnston, J., "Narratives: Twenty-five years later," *Topics in Language Disorders*, vol. 28, pp. 93–98, 2008. 1.2

[84] Kann, V., *On the approximability of NP-complete optimization problems.* Ph.D. dissertation, Royal Institute of Technology, Department of Numerical Analysis and Computing Science, 1992. 3.6.2

[85] Kasch, N. and Oates, T., "Mining script-like structures from the web," in *Proceedings of the NAACL-HLT 2010 Workshop on Formailism and Methodology for Learning by Reading*, 2010. 1

[86] Klein, D. and Manning, C., "Accurate unlexicalized parsing," in *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pp. 423–430, 2003. 3.3.1

[87] Kolodner, J. L., "An introduction to case-based reasoning," *Artificial Intelligence Review*, vol. 6, no. 1, pp. 3–34, 1992. 2.2.1

[88] Krueger, K. A. and Dayan, P., "Flexible shaping: how learning in small steps helps.," *Cognition*, vol. 110, pp. 380–394, 2009. 2.4.1

[89] Kuhn, H. W., "The Hungarian Method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955. 3.3.1

[90] Labov, W., *The Language of Life and Death*. Cambridge, U.K.: Cambridge University Press, 2013. 2.2.4

[91] Labov, W. and Waletzky, J., "Narrative analysis: Oral versions of personal experience," in *Essays on the Verbal and Visual Arts* (Helm, J., ed.), pp. 12–44, 1967. 2.2.4

[92] Lebowitz, M., "Planning stories," in *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, 1987. 1.3, 2.2.1

[93] Lee, G., Bulitko, V., and Ludvig, E., "Sports commentary recommendation system (SCoReS): Machine learning for automated narrative," in *Proceedings of the 8th AAAI AIIDE Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012. 1.2

[94] Lehnert, W., "Cognition, computers and car bombs: How Yale prepared me for the 1990s," in *Beliefs, reasoning and decision making* (Schank, R. and Langer, E., eds.), Hillsdale, NJ: Erlbaum, 1994. 3

[95] Leonard, S. A. and McClure, M., *Myth and Knowing: An Introduction to World Mythology*. McGraw-Hill, 2003. 1

[96] Levesque, H. J., "What is planning in the presence of sensing?," in *Proceedings of the 13th National Conference on Artificial Intelligence*, (Portland, Oregon), pp. 1139–1146, AAAI Press, 1996. 1.4.1

[97] Li, B. and Riedl, M. O., "An offline planning approach to game plotline adaptation," in *Proceedings of the 6th Conference on Artificial Intelligence for Interactive Digital Entertainment*, (Palo Alto, CA), pp. 45–50, AAAI Press, 2010. 1.2, 1.3, 2.2.1

[98] Li, B. and Riedl, M. O., "Creative gadget design in fictions: Generating novel object types in analogical spaces," in *Proceedings of the 8th ACM Conference on Creativity and Cognition*, (Atlanta, Georgia), pp. 41–50, 2011. 6.4.1

[99] Li, B. and Riedl, M. O., "A phone that cures your flu: Generating imaginary gadgets in fictions with planning and analogies," in *Proceedings of the 4th Workshop on Intelligent Narrative Technologies*, (Palo Alto, CA), pp. 41–48, AAAI Press, 2011. 2.2.1, 3.1, 3.10

[100] Li, B., Zook, A., Davis, N., and Riedl, M. O., "Goal-driven conceptual blending: A computational approach for creativity," in *Proceedings of the 2012 International Conference on Computational Creativity*, 2012. 4.7

[101] Li, B., Zook, A., Davis, N., and Riedl, M. O., "Goal-driven conceptual blending: A computational approach for creativity," in *Proceedings of the 3rd International Conference on Computational Creativity*, (Dublin, Ireland), 2012. 6.4.1

[102] Lieberman, H., Dinakar, K., and Jones, B., "Crowdsourced ethics with personalized story matching," in *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, (New York, NY, USA), pp. 709–714, ACM, 2013. 1.2

[103] Lin, D., Church, K., Ji, H., Sekine, S., Yarowsky, D., Bergsma, S., Patil, K., Pitler, E., Lathbury, R., Rao, V., Dalwani, K., and Narsale, S., "New tools for web-scale n-grams," in *Proceedings of the 7th International Conference on Language Resources and Evaluation*, (Valleta, Malta), 2010. 3.3.3, 3.3.3

[104] Lincoln, B., *Theorizing myth: Narrative, ideology, and scholarship*. University of Chicago Press, 1999. 1

[105] Lintean, M. C. and Rus, V., "Paraphrase identification using weighted dependencies and word semantics," *Informatica*, vol. 34, pp. 19–28, 2010. 3.3.1

[106] Liu, H. and Singh, P., "MAKEBELIEVE: Using commonsense to generate stories," in *Proceedings of the 18th National Conference on Artificial Intelligence*, 2002. 2.4.2

[107] Liu, J. and Birnbaum, L., "Localsavvy: aggregating local points of view about news issues," in *Proceedings of the 1st International Workshop on Location and the Web*, 2008. 1.2

[108] Lu, Y., Castellanos, M., Dayal, U., and Zhai, C., "Automatic construction of a context-aware sentiment lexicon: An optimization approach," in *Proceedings of the International World Wide Web Conference*, 2011. 2.3, 4.4.2

[109] Magerko, B., Manzoul, W., Riedl, M., Baumer, A., Fuller, D., Luther, K., and Pearce, C., "An empirical study of cognition and theatrical improvisation," in *Proceedings of the ACM Conference on Creativity and Cognition*, pp. 117–126, ACM, 2009. 2.1

[110] Maher, M. L., "Computational and collective creativity: Who's being creative?," in *Proceedings of The 3rd International Conference on Computational Creativity*, (Dublin, Ireland), pp. 67–71, 2012. 3

[111] Mairesse, F. and Walker, M., "Towards personality-based user adaptation: Psychologically informed stylistic language generation," *User Modeling and User-Adapted Interaction*, vol. 20, no. 3, pp. 227–278, 2010. 2.2.3

[112] Mar, R. A., Oatley, K., Djikic, M., and Mullin, J., "Emotion and narrative fiction: Interactive influences before, during, and after reading," *Cognition and Emotion*, vol. 25, pp. 818–833, 2011. [U3]

[113] Mateas, M. and Sengers, P., "Narrative intelligence," in *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*, (North Falmouth, MA), pp. 1–10, AAAI, 1999. 1, 1.2

[114] McIntyre, N. and Lapata, M., "Plot induction and evolutionary search for story generation," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1562–1572, 2010. 1.4, 2.5, 6.5

[115] Meehan, J., *The Metanovel: Writing Stories by Computers*. Ph.D. dissertation, Yale University, 1976. 1.3, 2.2.1

[116] Miller, G., "WordNet: A lexical database for english," *Communications of the Association for Computing Machinery*, vol. 38, pp. 39–41, 1995. 2.3, 3.3.1, 4.4.2

[117] Mohammad, S., "From once upon a time to happily ever after: Tracking emotions in novels and fairy tales," in *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, (Portland, OR, USA), pp. 105–114, Association for Computational Linguistics, June 2011. 2.2.4, 2.3

[118] Mohammad, S. M. and Turney, P. D., "Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon," in *Proceedings of the NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, (Stroudsburg, PA, USA), pp. 26–34, Association for Computational Linguistics, 2010. 2.3

[119] Montfort., N., *Generating Narrative Variation in Interactive Fiction*. Ph.D. dissertation, University of Pennsylvania, 2007. 1.2, 2.2.2

[120] Moorman, K. and Ram, A., *A Functional Theory of Creative Reading*. Technical report GIT-CC-94/01, College of Computing, Georgia Institute of Technology, 1994. 1.2, 2.2.4, 6.4.1

[121] Nelson, M. J. and Mateas, M., "Search-based drama management in the interactive fiction Anchorhead," in *Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, (Marina del Rey, California), June 2005. 3.1, 3.7

[122] Niehaus, J., *Cognitive Models of Discourse Comprehension for Narrative Generation*. Ph.D. dissertation, North Carolina State University, Department of Computer Science, 2009. 2.2.4

[123] Niehaus, J., Li, B., and Riedl, M. O., "Automated scenario adaptation in support of intelligent tutoring systems," in *Proceedings of the 24th Conference of the Florida Artificial Intelligence Research Society*, (Palm Beach, FL), 2011. 1.2

[124] Norman, W. T., "Toward an adequate taxonomy of personality attributes: Replicated factor structure in peer nomination personality ratings," *Journal of Abnormal and Social Psychology*, vol. 66, no. 6, p. 574583, 1963. 2.2.3

[125] Novick, L. and Cheng, P. W., "Assessing interactive causal influence," *Psychological Review*, vol. 111, no. 2, pp. 455–485, 2004. 3.10

[126] Oatley, K., "A taxonomy of the emotions of literary response and a theory of identification in fictional narrative," *Poetics*, vol. 23, pp. 53–74, 1994. [U3]

[127] Oleson, D., Sorokin, A., Laughlin, G. P., Hester, V., Le, J., and Biewald, L., "Programmatic gold: Targeted and scalable quality assurance in crowdsourcing," in *Proceedings of the 3rd Human Computation Workshop*, pp. 64–67, 2010. 3.2

[128] O'Neill, B. and Riedl, M. O., "Dramatis: A computational model of suspense," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014. 2.2.2, 2.2.4, 3.10

[129] Ontañón, S. and Zhu, J., "On the role of domain knowledge in analogy-based story generation," in *Proceedings of the 22nd International Joint Conferences on Artificial Intelligence*, (Barcelona, Spain), pp. 1717–1722, 2011. 2.2.1

[130] Orkin, J. D., *Collective Artificial Intelligence: Simulated Role-Playing from Crowdsourced Data*. Ph.D. dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2012. 2.5

[131] Ounis, I., Lioma, C., Macdonald, C., and Plachouras, V., "Research directions in Terrier," *Novatica/UPGRADE Special Issue on Web Information Access*, 2007. 2.5

[132] Ouyang, J. and Mckeown, K., "Towards automatic detection of narrative structure," in *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)* (Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., eds.), (Reykjavik, Iceland), European Language Resources Association (ELRA), May 2014. 2.2.4

[133] Pang, B. and Lee, L., "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, pp. 1–135, 2008. 2.3

[134] Pearl, J., "The foundations of causal inference," *Sociological Methodology*, vol. 40, p. 75149, 2010. 3.10

[135] Pennington, N. and Hastie, R., "The story model for juror decision making," in *Inside the Juror: The Psychology of Juror Decision Making* (Hastie, R., ed.), pp. 192–221, Cambridge, UK: Cambridge University Press, 1993. 1.2

[136] Pérez y Pérez, R. and Sharples, M., "MEXICA: A computer model of a cognitive account of creative writing," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 13, pp. 119–139, 2001. 1.3, 2.2.1

[137] Pérez y Pérez, R. and Sharples, M., "Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA," *Knowledge-Based Systems*, vol. 17, no. 1, pp. 15–29, 2004. 2.2.1

[138] Porteous, J., Cavazza, M., and Charles, F., "Narrative generation through characters point of view," in *The SIGCHI Conference on Human Factors in Computing Systems*, 2010. 2.2.2

[139] Porteous, J., Teutenberg, J., Charles, F., and Cavazza, M., "Controlling narrative time in interactive storytelling," in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems* (Tumer, Yolum, Sonenberg, and Stone, eds.), (Taipei, Taiwan), pp. 449–456, 2011. 2.2.1

[140] Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., and Webber, B., "The Penn Discourse Treebank 2.0," in *Proceedings of the 6th International Conference on Language Resources and Evaluation*, 2008. 2.4

[141] Prince, G., *A Dictionary of Narratology*. Lincoln, NE: University of Nebraska Press, 1987. 1.1, 1.1, 2.1, 2.1

[142] Propp, V. Y., *Morphology of the Folktale*. Austin, TX: University of Texas Press, 1968. Originally published in Russian in 1928. 2.2.1

[143] Purse, L., "The new spatial dynamics of the bullet-time effect," in *From Hollywood to "Reality" TV and Beyond*, pp. 151–160, 2005. 6.4.2

[144] Pustejovsky, J., Hanks, R., P. S., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., and Lazo, M., "The TIMEBANK corpus," in *Proceedings of Corpus Linguistics*, pp. 647–656, 2003. 2.4, 3.6

[145] Ram, A., *Question-Driven understanding: An integrated theory of story understanding, memory, and learning*. Ph.D. dissertation, Yale University, Department of Computer Science, 1989. Research Report No. 710. 2.2.4, 2.5

[146] Ram, A., "Creative conceptual change," in *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, (Boulder, CO), p. 1726, 1993. 6.4.1

[147] Regneri, M., Koller, A., and Pinkal, M., "Learning script knowledge with web experiments," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010. 2.4.2, 3.6

[148] Resnik, P., "Using information content to evaluate semantic similarity in a taxonomy," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 448–453, Morgan Kaufmann Publishers Inc, 1995. 3.3.1

[149] Riedl and León, "Generating story analogues," in *Proceedings of The 5th Conference on Artificial Intelligence in Interactive Digital Entertainment*, (Palo Alto, CA), 2009. 2.2.1

[150] Riedl, M. O., *Narrative Generation: Balancing Plot and Character*. Ph.D. dissertation, North Carolina State University, Department of Computer Science, 2004. 2.1.1

[151] Riedl, M. O., Rowe, J. P., and Elson, D. K., "Toward intelligent support of authoring machinima media content: Story and visualization," in *Proceedings of the 2nd International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN)*, (Playa del Carmen, Cancun, Mexico), 2008. 1.2

[152] Riedl, M. O., Stern, A., Dini, D., and Alderman, J., "Dynamic experience management in virtual worlds for entertainment, education, and training," *International Transactions on Systems Science and Applications*, vol. 3, no. 1, pp. 23–42, 2008. 1.4.1

[153] Riedl, M. O. and Sugandh, N., "Story planning with vignettes: Toward overcoming the content production bottleneck," in *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling*, (Erfurt, Germany), 2008. 1.3, 2.2.1

[154] Riedl, M. and Young, R., "Narrative planning: Balancing plot and character," *Journal of Artificial Intelligence Research*, vol. 39, pp. 217–268, 2010. 1.3, 2.2.1, 3.10

[155] Rishes, E., Lukin, S., Elson, D., and Walker, M., "Generating different story tellings from semantic representations of narrative," in *Proceedings of the 6th International Conference on Interactive Storytelling*, 2013. 2.2.3

[156] Rogelio E. Cardona-Rivera, Bradley A. Cassell, S. G. W. and Young, R. M., "Indexter: A computational model of the event-indexing situation model for characterizing narratives," in *the Proceedings of the 2012 Workshop on Computational Models of Narrative*, 2012. 2.2.4

[157] Rosch, E., "Cognitive representations of semantic categories," *Journal of Experimental Psychology: General*, vol. 104, no. 3, p. 192233, 1975. 1.1, 2.1

[158] Ross, S. M., *Introduction to Probability Models*. Academic Press, 11th ed., 2014. 4.3.1

[159] Rowe, J., Lobene, E., Mott, B., and Lester, J., "Play in the museum: Designing game-based learning environments for informal education settings," in *Proceedings of the 9th International Conference on the Foundations of Digital Games*, (Fort Lauderdale, FL), 2014. 1.2

[160] Sander, J., Qin, X., Lu, Z., Niu, N., and Kovarsky, A., "Automatic extraction of clusters from hierarchical clustering representations," in *Proceedings of the 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '03, (Berlin, Heidelberg), pp. 75–87, Springer-Verlag, 2003. 3.3.2

[161] Schank, R. and Abelson, R., *Scripts, plans, goals, and understanding: An inquiry into human knowledge structure*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1977. 1.3, 2.2.4

[162] Schmid, W., *Narratology: An Introduction*. Walter de Gruyter, 2010. 2.1, 2.1.1, 2.1.1

[163] Sharples, M., *How we write: Writing as creative design*. London, UK: Routledge, 1999. 1.2, 2.2.1

[164] Shklovsky, V., *Theory of Prose*. Walter de Gruyter, 1990. Originally published in 1929 in Russian. Translated by B. Sher. 2.1

[165] Si, M., Marsella, S. C., and Pynadath, D. V., "Thespian: Using multiagent fitting to craft interactive drama," in *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 21–28, 2005. 2.2.1

[166] Sina, S., Rosenfeld, A., and Kraus, S., "Generating content for scenario-based serious-games using crowdsourcing," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014. 1.4, 2.5, 6.5

[167] Singer, J. A., "Narrative identity and meaning making across the adult lifespan: An introduction," *Journal of Personality*, vol. 72, no. 3, pp. 437–460, 2004. 1.2

[168] Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., and Zhu, W. L., "Open Mind Common Sense: Knowledge acquisition from the general public," in *Confederated International Conferences DOA, CoopIS and ODBASE*, pp. 1223–1237, Springer-Verlag, 2002. 2.4.2

[169] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., and Potts, C., “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013. 2.3, 4.5.3.1

[170] Sparacino, F., “Museum intelligence: using interactive technologies for effective communication and storytelling in the puccini set designer exhibit,” 2004. 1.2

[171] Stanzel, F. K., *A Theory of Narrative*. Cambridge, England: Cambridge University Press, 1984. Translated by C. Goedsche. 2.1

[172] Stone, P., Dunphry, D. C., Smith, M. S., and Ogilvie, D. M., *The General Inquirer: A computer approach to content analysis*. Cambridge, MA: MIT Press, 1966. 2.3

[173] Strapparava, C. and Valitutti, A., “WordNet-Affect: an affective extension of WordNet,” in *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004. 2.3

[174] Strapparava, C., Valitutti, A., and Stock., O., “The affective weight of lexicon,” in *Proceedings of the 5th International Conference on Language Resources and Evaluation*, 2006. 2.3

[175] Strassen, V., “Gaussian elimination is not optimal,” *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969. 3.8

[176] Sunshine-Hill, B. and Badler, N., “Perceptually realistic behavior through alibi generation,” in *Proceedings of the 6th AAAI Conference on Artificial Intelligence for Interactive Digital Entertainment*, 2010. 6.3

[177] Swanson, R. and Gordon, A., “Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling,” *ACM Transactions on Interactive Intelligent Systems*, vol. 2, 2012. 1.4, 2.5, 6.5

[178] Swanson, R., Rahimtoroghi, E., Corcoran, T., and Walker, M., “Identifying narrative clause types in personal stories,” in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, (Philadelphia, PA, U.S.A.), pp. 171–180, Association for Computational Linguistics, June 2014. 2.2.4

[179] Swartjes, I., *Whose story is it anyway? How improv informs agency and authorship of emergent narrative*. Ph.D. dissertation, University of Twente, 2010. 2.2.1

[180] Talukdar, P. P., Wijaya, D., and Mitchell, T., “Acquiring temporal constraints between relations,” in *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM ’12, (Maui, Hawaii, USA), pp. 992–1001, ACM, 2012. 2.4.1, 2

[181] Tearse, B., Mawhorter, P., Mateas, M., and Wardrip-Fruin, N., "Skald: Minstrel reconstructed," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, pp. 156–165, June 2014. 2

[182] Todorov, T., "Categories of the literary narrative," *Film Reader*, vol. 2, pp. 19–37, 1977. 2.1, 2.1.1

[183] Tomai, E., *A Pragmatic Approach to Computational Narrative Understanding*. Ph.D. dissertation, Northwestern University, Department of Elecrtrical Engineering and Computer Science, 2009. Tech. Rep. No. NWU-EECS-09-17. 2.2.4

[184] Tomashevsky, B., "Thematics," in *Russian Formalist Criticism: Four Essays*, pp. 62–95, University of Nebraska, 1965. 2.1.1

[185] Tomashevsky, B. V., *Teoriya literatury. Poetika.* 1925. Reprint: Letchworth 1971. 2.1, 2.1.1

[186] Toutanova, K., Klein, D., Manning, C., and Singer, Y., "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the Human Language Technologies Conference*, NAACL-HLT '03, 2003. 4.4.2

[187] Trabasso, T. and Sperry, L., "Causal relatedness and importance of story events," *Journal of Memory and Language*, vol. 24, pp. 595–611, 1985. 1.4.1, 2.2.2

[188] Tupes, E. C. and Christal, R. E., "Recurrent personality factors based on trait ratings," Tech. Rep. USAF ASD No. 61-97, U. S. Air Force, Lackland Airforce Base, TX, 1961. 2.2.3

[189] Turner, M. and Fauconnier, G., *The Way We Think. Conceptual Blending and the Mind's Hidden Complexities.* New York: Basic Books, 2002. 2.2.4, 6.4.1

[190] Turner, S. R., *Minstrel: A Computer Model of Creativity and Storytelling.* Ph.D. dissertation, University of California at Los Angeles, 1993. 1.3, 2.2.1

[191] Turney, P. D. and Littman, M. L., "Measuring praise and criticism: Inference of semantic orientation from association," *ACM Transactions on Information Systems*, vol. 21, pp. 315–346, Oct. 2003. 2.3

[192] Valls-Vargas, J., Ontañón, S., and Zhu, J., "Toward character role assignment for natural language stories," in *Proceedings of the 6th Workshop on Intelligent Narrative Technologies*, 2013. 2.2.4

[193] van den Broek, P., "Discovering the cement of the universe: The development of event comprehension from childhood to adulthood," in *Developmental spans in event comprehension: Bridging fictional and actual events*, pp. 321–342, Mahwah, NJ: Lawrence Erlbaum, 1997. 1.3

[194] Vayanou, M., Katifori, A., Karvounis, M., Kyriakidi, M., Roussou, M., Tsangaris, M., Boile, M., Ioannidis, Y., Rennick-Egglestone, S., and Pujol, L., "The impact of interactive digital storytelling in cultural heritage sites," in *Proceedings of the International Digital Storytelling Conference: Digital Storytelling in Times of Crisis*, (Athens, Greece), 2014. 1.2

[195] Vilain, M., Burger, J., Aberdeen, J., Connolly, D., and Hirschman, L., "A model-theoretic coreference scoring scheme," in *Proceeding of the 6th Conference on Message Understanding*, pp. 45–52, 1995. 3.4

[196] Ware, S. G. and Young, R. M., "CPOCL: A narrative planner supporting conflict," in *Proceedings of The 7th Conference on Artificial Intelligence in Interactive Digital Entertainment*, (Palo Alto, CA), pp. 212–221, Oct 2011. 2.2.1

[197] Wauthier, F. and Jordan, M., "Bayesian bias mitigation for crowdsourcing," in *Advances in Neural Information Processing Systems 24 (NIPS)* (Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., eds.), pp. 1800–1808, 2011. 3.2

[198] Weltman, J. S., Iyengar, S. S., and Hegarty, M., "Mind the gap: Collecting commonsense data about simple experiences," in *Proceedings of the 2013 International Conference on Intelligence User Interfaces*, (Santa Monica, CA), pp. 41–48, AAAI Press, 2013. 2.4.2

[199] Weyhrauch, P., *Guiding Interactive Drama*. Ph.D. dissertation, Carnegie Mellon University, Department of Computer Science, 1978. Technical Report CMU-CS-97-109. 3.1

[200] Wilensky, R., *Planning and Understanding: A Computational Approach to Human Reasoning*. Reading, MA: Addison-Wesley Publishing Co., 1983. 1.3, 2.2.4

[201] Wilensky, R., "Why John married Mary: Understanding stories involving recurring goals," *Cognitive Science*, vol. 2, pp. 235–266, 1978. 3.10

[202] Williams, J. P., "Comprehension of students with and without learning disabilities: Identification of narrative themes and idiosyncratic text representations," *Journal of Educational Psychology*, vol. 85, pp. 631–641, 1993. 1.3

[203] Winston, P. H., "The strong story hypothesis and the directed perception hypothesis," in *The AAAI Fall Symposium*, 2011. 1, 1.2

[204] Wu, Z. and Palmer, M., "Verb semantics and lexical selection," in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics.*, p. 133138, 1994. 2.5, 3.3.1

[205] Young, R. M., Pollack, M. E., and Moore., J. D., “Decomposition and causality in partial-order planning,” in *The Proceedings of the Second International Conference on Artificial Intelligence and Planning Systems*, 1994. 2.2.1

[206] Zhu, J., Ontañón, S., and Lewter, B., “Representing game characters’ inner worlds through narrative perspectives,” in *The 6th International Conference on Foundations of Digital Games*, pp. 204–210, 2011. 2.2.2

[207] Zwaan, R. A. and Radvansky, G. A., “Situation models in language comprehension and memory,” *Psychological Bulletin*, vol. 123, no. 2, pp. 162–185, 1998. 4.3