11-2-2018

Automatic Extraction of Narrative Structure from Long Form Text

Joshua Daniel Eisenberg
Florida International University, jeise003@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

AUTOMATIC EXTRACTION OF NARRATIVE STRUCTURE
FROM LONG FORM TEXT

A dissertation submitted in partial fulfillment of the

Requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

By

Joshua Daniel Eisenberg

2018

To: Dean John Volakis
College of Engineering and Computing

This dissertation, written by Joshua Daniel Eisenberg, and entitled Automatic Extraction of Narrative Structure from Long Form Text, having been approved in respect to style and intellectual content, is referred to you for judgement.

Bogdan Carbunar

Shu-Ching Chen

Anthony Dick

Zhiming Zhao

Mark Finlayson, Major Professor

Date of Defense: November 2, 2018

The dissertation of Joshua Daniel Eisenberg is approved.

Dean John Volakis
College of Engineering and Computing

Andres G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2018

DEDICATION

I explore, research, and write, for you, Aimee.

I wouldn’t be half the person I am (and I’m not just speaking genetically) if it wasn’t for the love, support, and guidance of my parents.

ACKNOWLEDGMENTS

This work was supported by several federal grants. First, by the National Institutes of Health (NIH) grant number 5R01GM105033-02. Second, by the National Science Foundation (NSF) under Grant No. HRD-1547798, which was awarded to Florida International University as part of the Centers of Research Excellence in Science and Technology (CREST) Program. Third, by the NSF under Grant Award #1129076, via the Partnerships for International Research and Education (PIRE) Open Science Data Cloud (OSDC) fellowship. Fourth, the narrative levels annotation study was funded by the Office of Naval Research (ONR) Grant No. N00014-17-1-2983.

I want to thank my friend Jared. Without your advice, I would have never read the writing of Haruki Murakami. Thanks to you, I found Murakami's work, consumed thousands of pages of his stories, and was inspired to teach computers how to understand the interwoven and highly embedded narratives in his work.

I want to thank my advisor Mark Finlayson, for always providing the necessary dissents and criticism, which pushed me to find better problems and solutions. I also want to thank Mark for his encouragement, and always trying to give me the freedom to work on what I want, and not caring *where* I worked on it.

I want to thank W. Victor H. Yarlott for his help programming the feature extraction for the reimplemented story classifiers, and—more importantly—for always helping me brainstorm and come up with amazing, outlandish, and impractical ideas, which always end up being useful when I least expect it. I want to thank Deya, Labiba, Mireya, and Mohammed for the great conversations and experiences at the Indian buffet.

I would like to thank my annotators: Thanks to Fernando Serrano and Victor Alvarez for their work annotating the narrative point of view and diegesis of the first page of each book from the Corpus of English Novels. Thanks to Michael Alvarez and Guillermo Bonilla for their work annotating my narrative levels corpus. I couldn’t have taught my narrative level extractor programs if it wasn’t for your accurate annotations. A special thanks to Michael Alvarez for your in-depth insight into the usage of narrative levels, and for your help annotating the texts for the SANTA workshop.

ABSTRACT OF THE DISSERTATION

AUTOMATIC EXTRACTION OF NARRATIVE STRUCTURE FROM LONG FORM TEXT

by

Joshua Daniel Eisenberg

Florida International University, 2018

Miami, Florida

Professor Mark Finlayson, Major Professor

Automatic understanding of stories is a long-held goal of artificial intelligence and natural language processing research communities. Stories explain the human experience. Understanding stories promotes the understanding of both individuals and groups of people; cultures, societies, families, organizations, governments, and corporations, to name a few. People use stories to share information. Stories are told–by narrators–in linguistic bundles of words called narratives.

My work has given computers understanding of some aspects of narrative structure. Specifically, where are the boundaries of a narrative in a text. This is the task of determining where a narrative begins and ends, a non-trivial task, because people rarely tell one story at a time. People don’t specifically announce when we are starting or stopping our stories: We interrupt each other. We tell stories within stories. Before my work, computers had no awareness of narrative boundaries, essentially where stories begin and end. My programs can extract narrative boundaries from novels and short stories with an F_1 of 0.65.

Before this I worked on teaching computers to identify which paragraphs of text have story content, with an F_1 of 0.75 (which is state of the art). Additionally, I have taught computers to identify the narrative point of view (POV; how the narrator refers to theirself) and diegesis (how is the narrator involved in the story's action) with F_1 of over 0.90 for both narrative characteristics. For the narrative POV, diegesis, and narrative level extractors I ran annotation studies, with high agreement, that allowed me to teach computational models to identify structural elements of narrative through supervised machine learning.

My work has given computers the ability to find where stories begin and end in raw text. This will allow for further, automatic analysis, like extraction of plot, intent, event causality, and event coreference. These are difficult when there are multiple narratives in one text. There are two key contributions in my work: 1) my identification of features that accurately extract information about narrators, and narrative levels and 2) the gold-standard data generated from running annotation studies on identifying these same elements of narrative structure.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURES PAGE

ABBREVIATIONS AND ACRONYMS

ARD	Agreement relative to disagreement
ML	Machine learning
NER	Named entity recognizer
NLP	Natural language processing
PDF	Portable document format
POV	Narrative point of view
SANTA	Systematic analysis of narrative texts through annotation
SRL	Semantic role labeler
SVM	Support vector machine
TF-IDF	Term frequency-inverse document frequency
WSD	Word sense disambiguation

Chapter 1

Motivation

Stories explain the human experience. Understanding our stories promotes the understanding of both individuals and groups of people; cultures, societies, families, organizations, governments, and corporations, to name a few. People use stories to share information.

Stories are told–by narrators–in linguistic bundles of words called *narratives*. My work enables computers to have an awareness of the structural elements of narrative. In particular, computers can automatically determine where the boundaries of a narrative in a text are. This is the task of determining where a narrative begins and ends, a non-trivial task, because people often narrate multiple stories at a time. We don't specifically announce when we are starting or stopping our stories. We interrupt each other. We tell stories within stories. Before my work, we had no ability to automatically detect where stories begin and end in long texts (specifically texts between 350 and 1,000 sentences long or 7,000 to 20,000 words), which I will discuss further in this dissertation.

My programs can automatically extract structural information about narrative, and this gives information about how a story is told. Extraction of finer grained information, like plot, character roles, and event relationships, cannot be accurately extracted without knowing which spans of text contain the telling of which stories. Imagine trying to figure out the plot of a chapter from a novel, but there are two distinct narratives. Typically, there is an original narrative, which tells a story about a group of characters, and then one of the characters, possibly the protagonist, will tell a story about the past. Here are two distinct narratives, the original one, and the narrative embedded in the character's speech. Note that not all speech is considered narrative, so this task is more complex than just finding quoted text.

A classic example of this can be found in the novel "Heart of Darkness" by Joseph Conrad (2016). There are two main narratives in "Heart of Darkness": the original narrative begins with five men sitting on a boat, and one of these men, Marlow, is telling a story about his past. The second narrative, occurs in the speech of Marlow, which tells a story of his journey on the Congo River. The real action in "Heart of Darkness" occurs in the embedded narrative told by Marlow; his monologue to the other passengers of the boat. There is little action in the original narrative other than Marlow speaking, which provides a good example of an embedded narrative to annotate and examine.

What does the plot of "Heart of Darkness" look like? There are two distinct plots: first, a brief plot describing how Marlow tells his story to his four shipmates, and second, the plot of Marlow's past, going to Africa, managing a trading post, and going through the Congo to search for Mr. Kurtz (the mysterious antagonist). The embedded narrative is rich with events. How would a computer be able to extract these two plots, if it treats all text the same way, and assumes that there is only one narrative?

My work gives computers the ability to detect where narrative boundaries are, and how long are the spans of each narrative level. Using my narrative level extractor, computers can be aware of which spans of text belong to different narrative levels, and these different levels should each be treated as their own distinct narratives. In order to understand the narrative, the computer needs to extract plot, and it must extract plot from each narrative level. If the computer only tried to extract one plot from the entirety of "Heart of Darkness" then it would be convoluted, with events from different stories all smushed into the same plot. It is essential to distinguish different narrative levels, and treat them as unique narratives with their own plots, characters, and narrators.

Story understanding is an almost automatic awareness for people. According to cognitive psychologist Jerome Bruner (2003, pp. 8-9), "The telling of a story and its comprehension as a story depend on the human capacity to process knowledge in this interpretive way. …[T]here is compelling evidence to indicate that narrative comprehension is among the earliest powers of the mind to appear in the young child and among the most widely used forms of organizing human experience." My work is the first to teach computers the awareness of narrative point of view and diegesis, as well as extraction of narrative levels. My research has also advanced the ability of story classification, from F_1 of 0.65 to 0.75, using many orders of magnitude less features, and is thousands of times more generalizable.

People can automatically process stories, but computers must be programmed to have these skills. My work teaches computers the ability to process structural information from stories, what is the POV, what is the diegesis, is a story being told, and where are the boundaries between narratives. Further work can be done to determine basic components of story understanding, such as what is the plot of the stories that are told in each narrative, who are characters, do events in the story of one narrative get mentioned in the telling of another narrative. At the time of writing this dissertation, computers have rudimentary ability to solve these tasks. Further, an important aspect to each of these tasks, is being able to distinguish which spans of text belong to which narrative. These answers depend on where narrative levels occur, so automatic extraction of narrative levels is necessary for accurate extraction of plot, characters, and event-coreference for multi-narrative texts.

But why do computers need to understand stories? One reason is that stories will enable computers to understand people. If computers can understand the structure of stories, they can extract the information bundled into the story narrated by the person interacting with the computer. This information can be used to understand people based on their story, be it their history, their cultural narratives, or the narratives of the societal groups they belong. People use this information to understand other people, and computers should be aware of the people's narrative when they are interacting with us.

Second, story understanding can be used by computers to make models of what is happening in the real world in real time. Every day, new articles are published reporting on current events. People discuss these events on social media platforms like Twitter and Facebook. If computers could extract plots from the stories that are contained in news articles, then they could automatically make models of what is happening in the real world. It is important to know how people report and react to current events. If computers could find references to the events mentioned in the news, to the discourse posted on social media and the internet, they could obtain a sample of people's opinions on current issues.

My work will enable this type of story understanding because my work will allow the computer to find story content, specify which spans of text contain which narratives, and classify characteristics of the narrator. The applications for a system like this are endless. An obvious, and profitable, application of computational story understanding, is to automatically process the news, and social media, in real time, as a feature for making investment decisions. For example, if an article about Amazon's poor treatment of part-time workers is published, it might affect the behavior of the stock. Knowledge of how people react to this article is useful in considering how to interpret this news and its impact. Computational understanding of stories will allow computers to automatically extract this information, incorporate it into its understanding of the world, and use this knowledge to make more educated decisions.

If understanding people, cultural narrative, and enabling computers to make informed decisions aren't compelling reasons to research the computational understanding of stories, then consider the following: computational understanding of narrative could be used to automatically identify the spread of fake news. My programs can be used to figure out where stories begin and end in text. Once we know these spans, plot can be extracted, and the truth of these events, and relationships of events can be evaluated for facts. Without extraction of narrative levels, plot and event extraction would be carried out on too broad a scope.

1.1 Outline

The dissertation is organized as follows: Chapter 2 elucidates the elements of basic narratology, and narrative boundary theory. I define essential terms, including story, narrative, and narrative boundary. I will discuss plot and characters. I use examples from short stories, novels, and TV scripts to illustrate the usage of some common and not so common arrangements of embedded and interruptive narratives. In Chapter 3 I discuss the annotation study that I ran on the extraction of narrative boundaries in long form texts. I hired and trained two domain experts in narrative so that they could annotate the narrative boundaries in a corpus of 287,777 words, spread across 26 unique English texts: excerpts from novels by a diverse set of authors, short stories by Haruki Murakami, American court transcripts, and TV show screenplays. Chapter 4 contains the most significant result of this dissertation. Narrative boundaries can be accurately extracted from natural texts when computers use three key features: the narrative point of view, the usage of stories, and occurrence of main characters. These features allow an SVM model to achieve F_1 of 0.62 for detecting embedded narratives. Chapter 5 discusses the task of story classification in paragraphs of texts. I built a state of the art classifier, that is more accurate, generalizable, and computationally efficient with respect to the work of other researchers. The research in Chapter 6 is on the annotation of narrative diegesis and point of view from novels and news articles, and demonstrates the accurate computational extraction of these characteristics from natural texts. Chapter 7 is an exploration of the computational work that other researchers have conducted relating to the computational understanding of narrative. In Chapter 8 I conclude with summarizing my contributions and discussing future work and applications.

1.2 Dissertation contributions

There are two key sets of contributions in my dissertation:

1) The choice of which features are useful for automatically classifying different aspects of narrative. In chapter 4 I empirically show which features are best for extracting narrative levels from raw text. In chapter 5 I show which features are

best for detecting which paragraphs of text have story content. In chapter 6 I show which features are best for classifying the narrative point of view and diegesis of novels. Although engineering programs to extract these features, is a non-trivial task, knowing which features to use for classifying a narrative phenomenon is a much harder task. The feature engineering is the most important kernel of knowledge produced by my dissertation research. Going forward, any programmer, looking to teach their computers computational understanding of narrative, can use my findings to guide their implementations.

2) The secondary contributions are the gold-standard annotations produced from the annotation studies I ran, and the annotation guides that I wrote and used to run the annotation studies. It would be impossible to train and evaluate the machine learning models that classify elements of narrative structure without the annotated data produced in my annotation studies. It would also be impossible to determine which features can accurately classify the elements of narrative structure without these annotations. I include the annotation guides in this category of contribution, because these guides will help other people run their own annotation studies.

There is one final contribution that is worth mentioning: the Java (Gosling, 2014) programs that extract features, train SVM (Chang, 2011) classification models, and evaluate the performance of these models. The POV and diegesis extractors have already been open sourced and are available online[1]. The remainder of these programs can be shared by request[2]. The code is a secondary contribution; the features for how the computer should classify different aspects of narrative structure is key. Now anyone can use insights gained from my experiments on different sets of features to design more accurate and advanced systems that extract narrative structure.

In addition to the code being open sourced, I have been granted a patent for the "Features for the Automatic Classification of Narrative Point of View and Diegesis" (U.S. Patent No. 15/804,589), and I have a patent pending for the "Features for Classification of Stories" (U.S. Patent Application No. 62/728,380*)*. Hence, the main contribution of my work is the features used to classify narrative structure, and the relationship of these features to the realities of narrative. I did not patent my programs, or my algorithms. Instead I patented the features that the computer needs to make correct decisions.

[1] https://dspace.mit.edu/handle/1721.1/105279

[2] jeise003@fiu.edu

Chapter 2

Narrative theory

This chapter is a review of narratology, and the concepts that are essential for my research. A clear understanding of these narrative concepts is necessary for our discussions on extracting structural elements of narrative from text. I will first define the concepts of narrative, narrators, and story. Next, I will define two characteristics of narrators: point of view and diegesis. Then I discuss plot and characters. Finally, I will have a thorough discussion about narrative boundaries and levels, and explain how they can appear in real texts.

A by-product of explicitly defining and discussing the mechanics of these characteristics, is the ability to analyze which features of linguistics and semantics characterize them. This enabled me to make accurate decisions when doing my own annotations and when adjudicating annotations. My involvement in annotation POV, diegesis, and narrative boundaries enabled me to explain how I could decide the values for these characteristics. This was invaluable experience when it came time to decide which features to use for generating computational models.

The information in this chapter has been adapted from the annotation guides I wrote for my annotation studies on narrative boundaries, POV, and diegesis. I compiled this information to train my annotators. At a minimum, I wanted them to be aware of the narratological concepts that explain the phenomena they would be annotating. Ideally, I planned for my annotators to combine their *natural* ability to understand stories, with the theoretical frameworks that are defined in the annotation guides, to help them think critically, and if possible quantitatively, about how they made their decisions about narrative structure.

2.1 Narrative, narrators and stories

A narrative is a discourse presenting a coherent sequence of events which causally and purposely relate, concerns specific characters and times, and overall displays a level of organization beyond the commonsense coherence of the events themselves, such as that provided by a climax or other plot structure.

Narrative is a linguistic representation of a *story*. A *story* is a series of events affected by animate actors or characters. A story is an abstract construct, with two essential elements: plot (*fabula)* (Bal, 2009, p. 5) and characters (*dramatis personae*) (Propp, 1968, p. 20). The art of storytelling is much more complicated than merely listing events carried out by characters. There is great importance in the storyteller's choice of which details are revealed to the reader, the order in which plot events are told, whether to embed stories within each other, and whether to interrupt the telling of one story to make space for a new one. Even the choice of what details (character traits, setting, history) the author reveals to the reader is important.

Narrative is more concrete than story, in that narrative is made up of words, but a story is formed through the co-occurrence of characters who enact events which advance a plot forward. Narratives occupy spans of text, while stories are a more complex relationship involving characters and events. Throughout this dissertation, I will say the *narrative* is the span of text that expresses the story, or simply *narrative text*. Note, narratives can also be delivered in the form of images, movies, and songs. In my dissertation, I focus on text narratives, but I do some work on automatic understand of TV show scripts, which is discussed in chapters 3 and 4.

Narrators tell narratives. "A narrative is narrated by a narrator to a narratee…" (Nelles, 1997, p. 9). A narrator is not the same as an author. In narratology, there is a framework of historical authors and readers, and implied authors and readers (Bal, 2009, p. 16). The historical author writes the narrative, while the historical reader either reads or hears the narrative. An implied author is closer to the concept of narrator: the implied author is the entity, with respect to the frame of the narrative, that is narrating, or telling the story. The implied author only exists within the text. The implied reader is the entity that the text is being read or written to. "The historical author writes, the history reader reads; the implied author means, the implied reader interprets; the narrator speaks, the narrate hears." (Nelles, 1997, p. 9).

Narrative diegesis is whether the narrator is involved or not involved in the story's action, heterodiegetic or homodiegetic, respectively. In a homodiegetic narrative, the narrator is not just a narrator, but a character as well, performing actions that drive the plot forward. In a heterodiegetic narrative, the narrator is observing the action but not influencing its course.

Narrators can tell stories in different ways and there are different types of narrators; different characteristics in which the narrator tells a story. The *point of view* (POV) of a narrator is whether the narrator describes events in a personal or impersonal manner. There are, in theory, three possible points of view, corresponding to grammatical person: first, second, and third person. First person point of view involves a narrator referring to themself, and implies a direct, personal observation of events. By contrast, in a third person narrative the narrator is outside the story's course of action, looking in. The narrator tells the reader what happens to the characters of the story without ever referring to the narrator's own thoughts or feelings.

2.2 Narrative levels in literature

Narratives can be arranged in many interesting ways: a narrative can appear contiguously as one solid span of text, or it might be embedded in another narrative, or it might even interrupt the preceding narrative. There are infinite ways to arrange embedded and interruptive narratives. An embedded narrative can be interrupted. An interrupted narrative can have embedded narratives within it. Embedded and interruptive levels can be used by storytellers in any arrangement of their choice. Many novels and short stories contain multiple instances of embedded and interruptive narratives, often with intricate combinations of the two phenomena. This is also true of scripts of TV shows, movies and the transcripts of court cases.

Every narrative has at least two narrative boundaries: the **start point**—which I define here as the position in text of the first character of the first word in the narration—and the **end point**—which I define as the position of the last character after the last word in the narration. The simplest kind of narrative is an uninterrupted one. The start point of such a narration is the first character of the text, and the end point is the last character of the text. This text's narrative has only two boundaries.

> I woke up early in the morning, checked the weather app on my phone and decided it would be a perfect day to go to the beach. I grabbed a book, a towel, and sunglasses, got in my car, and drove to the beach. I read my book, watched the waves, and went for a quick swim. I dried off and drove home. It was a great day, even though I forgot to bring sun screen and got a sunburn.

Example 2.1: A simple example of uninterrupted narrative

Example 2.1 shows an uninterrupted narrative by a first-person narrator who tells the story of their trip to the beach. The narrator uses the first-person point of view to narrate. There are no shifts in time, and no interrupted narratives. The next two sections define embedded (§ 2.2.1) and interruptive (§ 2.2.2) narratives.

2.2.1 Embedded narratives

Narratives can be embedded in one another. An embedded narrative tells a story within a story. Before I discuss how embedded narratives occur in text, let's define how I refer to the relationship between the layers. The **original narrative** is the narrative where the embedded narrative is told, and the original narrative contains an event (explicit or implied) that signals the telling of an embedded narrative. The **embedded narrative** is the narrative that is embedded within the original narrative.

Figure 2.1 contains a narrative level diagram for a text that contains an embedded narrative. The lower bar represents the span of text that the original narrative appears in, while the upper bar represents the spans of the embedded narrative. The horizontal axis represents the text under consideration. Progressing from left to right, the graph represents the position in the text advancing from the first word to the last. The horizontal axis can sometimes represent the progression of time, but time is not always linear in a narrative; there can be flashforwards, flashbacks, and other anachronisms.

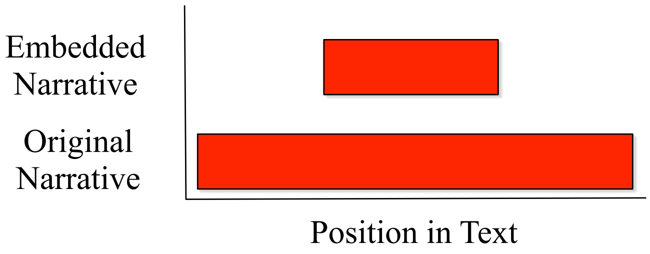


Figure 2.1 Narrative level diagram for an embedded narrative.

An embedded narrative occurs when a plot event in the original narrative triggers the telling of another story in the story. The narrative that tells the second story is the embedded narrative. A common example of embedded narrative is a conversation that occurs in the original narrative. Within the dialogue, one (or both) of the participants narrates a story. The plot event in the original narrative that signals the embedded narrative is a character's narration of a story via speech. This speaking is an event occurring on the original level. The events in the character spoken narrative belong to the plot of the story from the embedded narrative.

Recall Example 2.1, a story about a day at the beach. Example 2.2, is an altered version of Example 2.1, with the insertion of an embedded narrative. The span of text that contains the embedded narrative is highlighted in green. The narrative boundaries in Example 2.2 are graphically represented in the narrative boundaries diagram in Figure 2.1.

I woke up early in the morning, checked my weather app on my phone and decided it would be a perfect day to go to the beach. I grabbed a book, a towel, and sunglasses, got in my car, and drove to the beach. I read my book, watched the waves, and went for a quick swim. As I emerged from the water a disheveled looking pirate washed ashore, "Aye Aye! I have just been washed ashore. I was the captain of the Shivering Sparrow, but there was a mutiny onboard. All of my crew including my parrot turned on me, and made me walk the plank. I clung onto a piece of driftwood for three days, and now I am here. Where am I?" I didn't believe the pirate's story, so I ignored him and walked away. I dried off and drove home. It was a great day, even though I forgot to bring sunscreen and got a sunburn.

Example 2.2: Simple example of an embedded narrative

Example 2.2 contains a basic example of an embedded narrative. The story is almost the same as Example 2.1's, except when the original narrator gets out of the water, they encounter a pirate, who tells their story about being abandoned at sea, clinging to a piece of wood, and washing ashore. The pirate's embedded narrative is highlighted in green, and is embedded in the narrative of the original narrator. The original narrative ends the same way in Example 2.1.

In Example 2.2's original narrative, the original narrator witnesses the pirate telling a story. The pirate's narration is a plot event in the lower level. This plot event in the original narrative triggers the start of the embedded narrative. The plot events of the pirate's story are part of the embedded story, since they are told in the pirate's embedded narrative. The embedded narrative contains a story with events that are separate from the events in the story from the original narrative. It is also possible for the original narrator to tell an embedded narrative in the narrative text[3]. This type of narrative can occur via embedded flashbacks, which will be discussed in §2.2.3.

Example 2.3 is an excerpt from a novel. This excerpt contains an embedded narrative, highlighted in green. In the original narrative, Tsukuru Tazaki and Sara are on a date at a bar, and Tsukuru is telling Sara a story about his past. The highlighted text is part of the embedded narrative, since it contains Tsukuru narrating a story about his previous rejection.

This is a narration about Tsukuru Tazaki's past; He is explaining how he felt, and why he acted a certain way. Only the highlighted text is part of the embedded narrative. The final paragraph is not part of the embedded narrative because it is not a telling of the embedded story. It is part of the original narrative, where Sara is trying to verbalize her empathy for Tsukuru by asking him a clarifying question.

[3] Here *narrative text* means the text the narrator uses to narrate to the reader. Narrative text does not include text in quotes or direct speech.

> Her mojito glass was empty. She signaled the bartender and asked for a wine list, and, after some deliberation, she chose a glass of Napa Cabernet Sauvignon. Tsukuru had only drunk half his highball. The ice had melted, forming droplets on the outside of his glass. The paper coaster was wet and swollen.
>
> "That was the first time in my life that anyone had rejected me so completely," Tsukuru said. "And the ones who did it were the people I trusted the most, my four best friends in the world. I was so close to them that they had been like an extension of my own body. Searching for the reason, or correcting a misunderstanding, was beyond me. I was simply, and utterly, in shock. So much so that I thought I might never recover. It felt like something inside me had snapped."
>
> The bartender brought over the glass of wine and replenished the bowl of nuts. Once he'd left, Sara turned to Tsukuru.
>
> "I've never experienced that myself, but I think I can imagine how stunned you must have been. I understand that you couldn't recover from it quickly. But still, after time had passed and the shock had worn off, wasn't there something you could have done? I mean, it was so unfair. Why didn't you challenge it? I don't see how you could stand it."

Example 2.3 Example of an embedded narrative from a novel (Murakami 2014, p.41).

It is important to note that the phrase "Tsukuru said" in the second paragraph is not part of the embedded narrative because it is an action that occurs in the original narrative. Tsukuru is having his conversation within the frame of the original narrative while he is on a date with Sara.

Before I move to interruptive narratives, let’s talk about a canonical example of an embedded narrative: Joseph Conrad’s “Heart of Darkness” (Conrad, 2016). In this novel, there is a homodiegetic narrator on a boat, listening to a story told by his shipmate Marlow. Marlow’s story, which is told in dialogue, is the main story of the novel. The original narrator’s story is quite simple, he is just a passenger on a boat listening to Marlow. The story with action, and a compelling plot, is the embedded story that Marlow is telling the original narrator, about Marlow’s experiences in Africa, searching for Mr. Kurtz on the Congo.

2.2.2 Interruptive narratives

Narratives that interrupt the original narrator’s narration are called **interruptive narratives**, which are different from embedded narratives. An example is a book where each chapter has a different narrator. Namely, for the majority of the novel *1Q84* (Murakami, 2011), all the odd numbered chapters are narrated from the perspective of the heroine, Aomame, and the even numbered chapters are narrated from the perspective of the hero, Tengo. The boundaries at the end of each chapter in this novel mark interruptive narrative boundaries. For example, at the end of an odd numbered chapter, the narrator switches from the perspective of Aomame to Tengo, and at the end of each even numbered chapter, the narrator switches from the perspective of Aomame to Tengo.

Interruptive narratives can occur within chapters, or, for our purposes, within short stories, chapters of novels, or in the dialogue of a script. Sometimes the person narrating will change; at other times, the original narrator is a first-person narrator, and then the narrator will suddenly shift to a third person impersonal narrator, or vice versa. If the narrator changes, there is usually an interruptive narrative boundary.

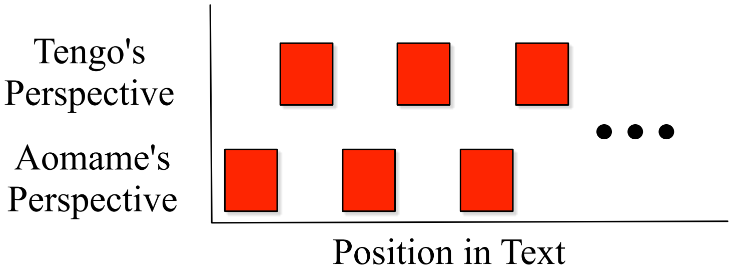


Figure 2.2 Narrative boundary diagram for "1Q84".

Sometimes there will be a section break which indicates the change of narrator. Section breaks are visual markers that separate text. Sometimes a section break is signaled by a series of special characters, like an asterisk (*) or a horizontal rule (a thin, horizontal line). Sometimes there will just be many blank lines in a section break. Note that the presence of a section break does not guarantee the presence of an interrupted narrative. For example, there can be a section break, and immediately after the break the narration is continued by the same narrator, from the same point in time that the narrative before the section break left off.

The difference between interruptive and embedded narratives may seem subtle, but there is a difference. In an embedded narrative, a plot event occurs in the story of the original narrative, which triggers the telling of an embedded narrative. An interruptive narrative is triggered by the original narrative stopping. The trigger of an interruptive narrative is not a plot event in the original narrative, instead there is a meta-event, where something structural, where how story is being told, changes. Once the original narrative has stopped, the interrupting narrative begins to be told. Even though the entity narrating the story can change, or the narrator remains consistent, the time in which the story is told changes. If you are questioning whether a narrative is interruptive, you should ask yourself: Is the telling of the span in question a plot event in the original narrative? If it is, then the span in question is embedded. If not, then it is interruptive.

I woke up early in the morning, checked my weather app on my phone and decided it would be a perfect day to go to the beach. I grabbed a book, a towel, and sunglasses, got in my car, and drove to the beach. I read my book, watched the waves, and went for a quick swim. As I emerged from the water a disheveled looking pirate washed ashore.

* * *

I have just been washed ashore. I was the captain of the Shivering Sparrow, but there was a mutiny onboard. All of my crew including my parrot turned on me, and made me walk the plank. I clung onto a piece of driftwood for three days, and now I am here.

* * *

The pirate looked like he just went through a tragic ordeal, but he was a pirate, so I decided it was best to ignore him. I dried off and drove home. It was a great day, even though I forgot to bring sun screen and got a sunburn.

Example 2.4: Example of an interruptive narrative

Let’s consider an example of a story with an interruptive narrative. Above is Example 2.4. It is again an altered version of Examples 2.1 and 2.2. The story is like Example 2.2, in that the narrator goes to the beach, reads, goes for a swim, and encounters a pirate upon exiting the water. After the original narrator observes the pirate washing ashore, there is a section break signaled by three asterisks. Highlighted in yellow is the interruptive narrative of the pirate, told in first person. The pirate telling this story is not an event in the original narrative, which is what happened in the embedded narrative of Example 2.2. There is no event, in the original narrative of Example 2.4, where the pirate tells a story. Instead, there is an interruption of the original narrative, the pirate tells his story, and then the original narrator begins telling his story. Figure 2.3 contains a narrative boundary diagram for this generic interruptive narrative.

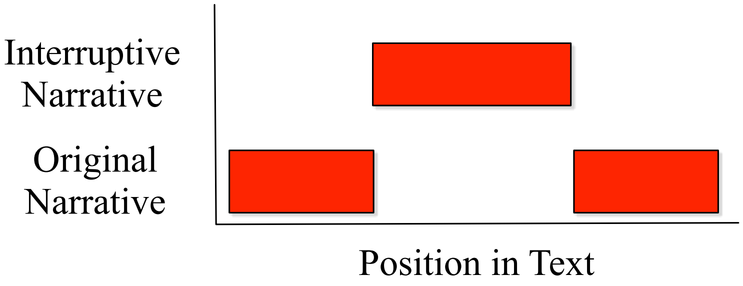


Figure 2.3: Narrative level diagram for an interruptive narrative

A useful set of questions to ask when looking for interruptive narratives include: who is the narrator? Who is the narratee? And, does this change over the course of the text? In an interrupted narrative, there is usually a change to either who the implied narrator is or the narratee, or both.

Now let's ask these questions about Example 2.4. The original narrative is narrated by the unnamed character who spends their day at the beach. The interruptive narrative is told by a new narrator: the pirate. It is unclear whether the narratee changes in these two narratives. The interruptive narrative could just be the pirate narrating his stream of consciousness to himself. There is not enough information to definitively say who the narratees are and whether they change. What is certain, is that the narrator changes during this interruptive narrator, and this is just another indication that the pirate's story is told in an interrupted narrative.

“You can hide memories, but you can’t erase the history that produced them.” Sara looked directly into his eyes. “If nothing else, you need to remember that. You can’t erase history, or change it. It would be like destroying yourself.”

“Why are we talking about this?” Tsukuru said, half to himself, trying to sound upbeat. “I’ve never talked to anybody about this before, and never planned to.”

Sara smiled faintly. “Maybe you needed to talk with somebody. More than you ever imagined.”

• • •

That summer, after he returned to Tokyo from Nagoya, Tsukuru was transfixed by the odd sensation that, physically, he was being completely transformed. Colors he’d once seen appeared completely different, as if they’d been covered by a special filter. He heard sounds that he’d never heard before, and couldn’t make out other noises that had always been familiar. When he moved, he felt clumsy and awkward, as if gravity were shifting around him.

For the five months after he returned to Tokyo, Tsukuru lived at death’s door. He set up a tiny place to dwell, all by himself, on the rim of a dark abyss. A perilous spot, teetering on the edge, where, if he rolled over in his sleep, he might plunge into the depth of the void. Yet he wasn’t afraid. All he thought about was how easy it would be to fall in.

Example 2.5: Example of an interruptive narrative from a novel (Murakami 2014, p. 45).

Next, let’s will consider another excerpt from Murakami (2014). Example 2.5 contains two narratives, highlighted in red and yellow. The first narrative, highlighted in red, is a continuation of the original narrative from Example 2.3, when Tsukuru is on a date with

Sara. The first narrative is interrupted by a third person narrator, who tells a story about Tsukuru's adolescence. There is a narrative break punctuating the two narratives. The narrative highlighted in yellow is an instance of an interruptive flashback, which is discussed in the next section.

2.2.3 Time shifts: flashbacks and flashforwards

There are two types of time shifts in story telling: flashbacks, (also known as analepsis), and flashforwards (also known as prolepsis). Both flashbacks and flashforwards are recurrent in storytelling. A flashback occurs when the time of the events told in the narration shift from the present to a time in the past. Flashbacks might occur when the narrator remembers something that happened in the past. A flashforward is similar, except the events are from the future. Flashforwards can come in the form of visions or prophecies. Other times, flashforwards foreshadow or reveal key events that will occur, even though the narrator might not know that these events will occur. Both flashbacks and flashforwards are popular storytelling devices in both literature and film. There are two ways flashbacks can be narrated:

Embedded flashbacks are embedded in the original narrative. In the original narrative, the narrator is narrating a story about the present, and then the narrator will shift the subject of their narration to telling a story about events that happened in the past. Sometimes the retelling of past events will use verbs in the past tense. The narrator is telling a story about the past from the present time, in which the events of the original narrative are unfolding. This is similar to the case where an embedded narrative is told in dialogue (see Example 2.2), except in flashbacks the embedded narrative is told in the narrative text; the audience of the flashback is the reader, not another character in the story.

Interruptive flashbacks interrupt or replace the original narrative. The original narrative ends, and a new narrative of events occurring at a time before the original narrative begins. The key characteristic of the interruptive flashback, is that the narrator also shifts in time. The narrator of the original narrative and the flashback do not have to be the same narrator. Sometimes the person who is narrating the flashback will be a different character than the narrator of the original narrative. Sometimes the point of view of the flashback's narrator will be different than that of the original narrator. Other times, the narrator of the flashback is identical to the original narrator, the only difference being the events in the flashback happened at a time before the original narrative. Interruptive flashbacks break the telling of the original narrative: they are not embedded in any other narrative.

Remember that the excerpt in Example 2.5 contains an interruptive flashback. The original narrative is interrupted by a new narrative, which takes place at a time before the original narrative. The narrator seems to be the same third person heterodiegetic narrator, but they are telling events from a story happened in a time prior to the events in the original narration.

Flashforwards can also either be embedded or interruptive. Flashforwards tend to be interruptive though, since narrators typically do not know what will happen in the future, so the original narrative must be interrupted, to provide an account of events from a future time. Flashbacks can be embedded into speech, but this is usually either a telling of a vision, or it can be the telling of a hypothetical future.

Flashbacks and flashforwards have elements of narrative structure that a simple mentioning of events from a different time lack. A time shift should have its own plot, characters, and story.   Flashbacks and flashforwards are stories and not just details from an event that happened in the past or future.

2.2.4 Dreams and visions

Many stories contain dreams. There are two types of dreams, and they are similar to the two types of flashbacks. Dreams are either embedded into the original narrative, or they interrupt it. Embedded dreams occur when the narrator is narrating the memory of their experience of a past dream. Interruptive dreams occur when the narration is occurring from *within* the dream: the narrator is narrating as the dream unfolds.

Visions are similar to dreams. A vision, like a prophecy, could be a telling of the future. The events of the prophetic vision may or may not come true, but the actual telling of the vision is distinct from the original narrative. Other types of visions can be sudden recollections of images or events from the past. Like dreams, visions and prophecies can be either embedded in the original narrative, or interruptive of the original narrative.

The excerpt in Example 2.6 is also from “Colorless Tsukuru Tazaki...”(Murakami 2014). The original narrative is highlighted in green. This narrative is about Tsukuru talking to his friend Haida about classical music. Talking about classical music causes Tsukuru to have a vision, or a daydream, from his past. In Tsukuru’s vision, which is highlighted in green, he sees his old friend Shiro masterfully playing the piano in a very dreamy and vivid setting.

Haida got quite talkative when it came to music. He went on, delineating the special characteristics of Berman's performance of Liszt, but Tsukuru barely listened. Instead, a picture of Shiro performing the piece, a mental image, vivid and three-dimensional, welled up in his mind. As if those beautiful moments were steadily swimming back, through a waterway, against the legitimate pressure of time.

The Yamaha grand piano in the living room of her house. Reflecting Shiro's conscientiousness, it was always perfectly tuned. The lustrous exterior without a single smudge or fingerprint to mar its luster. The afternoon light filtering in through the window. Shadows cast in the garden by the cypress trees. The lace curtain wavering in the breeze. Teacups on the table. Her black hair, neatly tied back, her expression intent as she gazed at the score. Her ten long, lovely fingers on the keyboard. Her legs, as they precisely depressed the pedals, possessed a hidden strength that seemed unimaginable in other situations. Her calves were like glazed porcelain, white and smooth. Whenever she was asked to play something, this piece was the one she most often chose. "Le mal du pays." The groundless sadness called forth in a person's heart by a pastoral landscape. Homesickness. Melancholy.

As he lightly shut his eyes and gave himself up to the music, Tsukuru felt his chest tighten with a disconsolate, stifling feeling, as if, before he'd realized it, he'd swallowed a hard lump of cloud. The piece ended and went on to the next track, but he said nothing, simply allowing those scenes to wash over him. Haida shot him an occasional glance."

Example 2.6: Example of an interruptive vision in a novel (Murakami 2014, pp. 70–71).

This vision interrupts the story told in the original narrative. The vision is not embedded because there is no action in the original narrative that triggers the telling of the vision. The last two sentences of the first paragraph inform the reader that Tsukuru is about to have a vision. These preparatory sentences are not part of the vision, since they describe events that are happening in the original narrative level: a "picture of Shiro…welled up in [Tsukuru's] mind." The actual vision is a departure from the original narrative. It describes what Tsukuru sees and feels when he is watching Shiro at the piano. This is not something that is *happening* at the time of the original narrative, it is something that Tsukuru is *experiencing.* The vision ends when the original third person narrator begins narrating about events that are actually happening in the present, "As he lightly shut his eyes and gave himself up to the music, Tsukuru felt his chest tighten with a disconsolate, stifling feeling…". The music then continues to play, and Haida shoots Tsukuru "…an occasional glance." These are events happening within the boundaries of the original narrative, and they signal the switch back to the original narrative from the interruptive vision.

2.3 Narrative levels in scripts

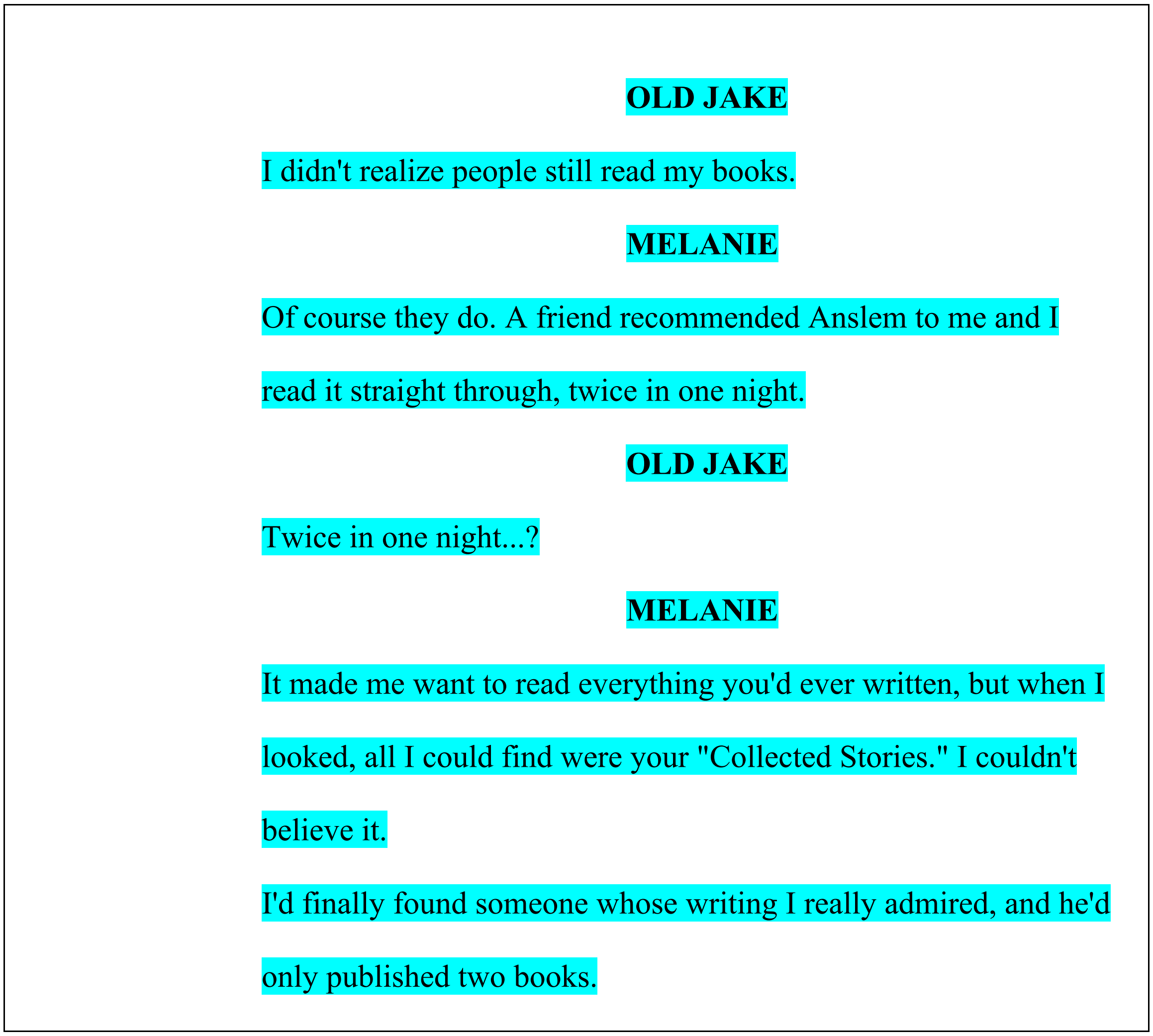
In addition to short stories and novels, I am interested in annotating the narrative boundaries in scripts. Specifically, this dissertation will focus on the scripts of TV shows, and the transcripts of court proceedings. There are two types of text in a script: dialogue and action. Dialogue contains the words that actors (or people) speak, and the action gives direction for what the actors do, how they do it, and what happens in the world that the script describes. Scripts can either be used to tell actors what to say and how they should act, which dictates how they should behave during a performance, or scripts can be a recording of things that happened in real life, like a transcription of the dialogue in a court case.

2.3.1 Dialogue

In the context of scripts, dialogue is a type of structured text. There are two components of dialogue: the **character's name** and the **character's speech.** In a script, the character's name will be stated. Typically, it will be bolded. Following the character's name are the words that the character will speak. The character's speech will not be in bold. Look at Example 2.7, which is an excerpt from the script of "Star Trek: Deep Space 9 – The Visitor" *(Taylor 1995)*. This excerpt portrays a conversation between two characters, Old Jake and Melanie. They are having a conversation about Old Jake's writings and how Melanie enjoys his writing. In this excerpt, there are four utterances in the dialogue. Old Jake speaks first, Melanie speaks next, and then they each speak one more time.

Now let's think about the script of this conversation with respect to the narrative boundaries it contains. There are two narratives. The original narrative, where Old Jake and Melanie are having a conversation. This narrative makes up the entire span of text in Example 2.7. The span of the original narrative has been highlighted in blue.

It is important to note that the bolded character names have also been highlighted. The character names belong to the original narrative because this is a signal that a specific character will utter the proceeding text. The declaration of who is speaking in a script is like the phrase “He said…” or “Old Jake said…” in a novel or short story. The character names are highlighted in the original narratives since they mark the beginning of a character speaking, which is an action in the original narrative.



OLD JAKE

I didn't realize people still read my books.

MELANIE

Of course they do. A friend recommended Anslem to me and I read it straight through, twice in one night.

OLD JAKE

Twice in one night...?

MELANIE

It made me want to read everything you'd ever written, but when I looked, all I could find were your "Collected Stories." I couldn't believe it.

I'd finally found someone whose writing I really admired, and he'd only published two books.

Example 2.7: An excerpt from “Star Trek: Deep Space 9 – The Visitor” with dialogue highlighted.

This excerpt from “Star Trek” (Taylor, 1995) also contains an embedded narrative delivered by Melanie. The embedded narrative is highlighted in yellow in Example 2.8. Her narration is about her experience reading Old Jake’s books, and how she reacted to his writing. In this embedded narrative, the bolded character names are not highlighted. This is because the action of Melanie speaking belongs to the plot of the original narrative, and they do not belong to the plot of the narrative about Melanie’s past. It is important to notice that Old Jake’s speech is not part of the embedded narrative: he is not adding any information to the story of Melanie’s past, he’s just asking a clarifying question.

OLD JAKE

I didn't realize people still read my books.

MELANIE

Of course they do. A friend recommended Anslem to me and I read it straight through, twice in one night.

OLD JAKE

Twice in one night...?

MELANIE

It made me want to read everything you'd ever written, but when I looked, all I could find were your “Collected Stories.” I couldn't believe it. I'd finally found someone whose writing I really admired, and he'd only published two books.

Example 2.8: An excerpt from “Star Trek: Deep Space 9 – The Visitor” with an embedded narrative in highlights.

2.3.2 Action

The action describes what is happening in the world that the script depicts. Typically, the action is written in present tense, since it describes what is happening in the present moment. Dialogue prescribes what each character says, and action dictates what each character does, including the way they speak. Consider Example 2.9, where the action is highlighted in green. Typically, the action in a script will be bolded, but it is not a requirement.

Now I will discuss the functions of each action sequence. The first sequence describes actions that Jake does before he speaks. The second action is during Jake's dialogue. It is a note for the actor playing Jake to take a moment to *consider* what he is saying. If the script is being read, then this stage direction allows the reader to imagine the character considering their actions. The third action sequence describes how Melanie reacts to what Jake says, and how she responds to him. The fourth action sequence instructs Melanie's next line to be said *softly.* The final action sequence describes an action Jake takes.

All five of these actions sequences describe actions that occur in the original narrative of this script. When considering the narrative boundaries for this excerpt, each action sequence is a part of the original narrative. In fact, the entire span of text in Example 9 belongs to the original narrative. There are no embedded or interruptive narratives in this excerpt.

MELANIE

So that I could read them again... like it was the first time.

Jake smiles, nods that he understands. As he sits down with the tray...

OLD JAKE

There's only one "first time" for everything, isn't there?

(considers)

And only one last time, too. You think about that when you
get to be my age. That today might be the last time you...
sit in a favorite chair... watch the rain fall... enjoy a cup of tea.

Melanie looks at him, then cautiously asks the question that brought her here.

MELANIE

(softly)

Can I ask you something... ?

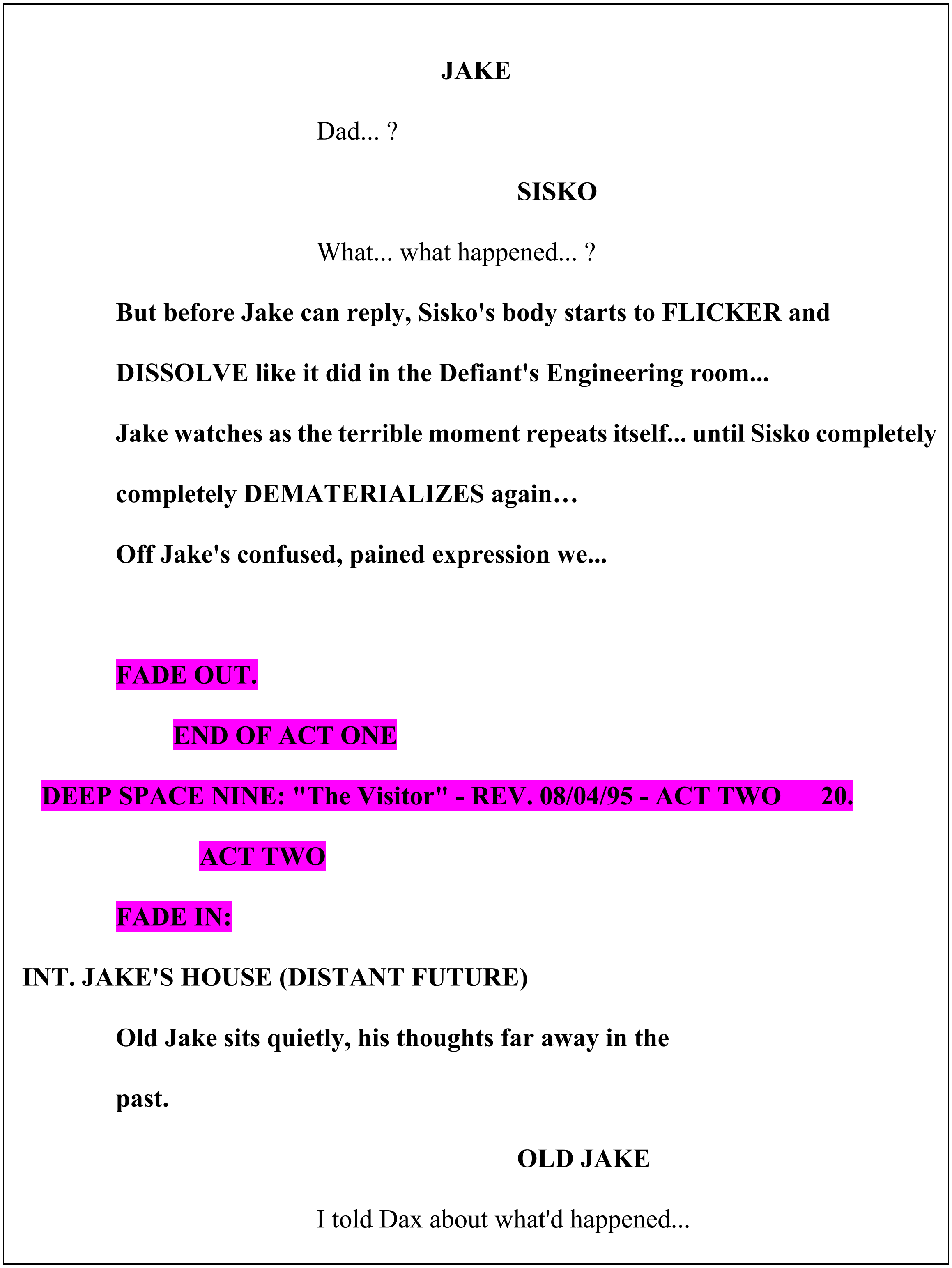
He nods that she go ahead...

Example 2.9: An excerpt from "Star Trek: Deep Space 9 – The Visitor" highlighted in green to distinguish action.

2.3.3 Structural elements of scripts

Structural elements are a final component of scripts that are separate from action and dialogue. They allow the readers or actors to distinguish between scenes and acts, and they give notes about the technical production for the performance, like a change of a camera angle. In the study, I do not include structural elements in our narratives. These elements are not part of the story being told, they just instruct the actors and crew when a scene begins or ends, and tell the camera operators logistics for how the scene is shot.

Example 2.10 is an excerpt from Star Trek: Deep Space 9 (Taylor, 1995). It has the structural elements highlighted in pink. In this example, the structural elements prescribe the camera fading out, the first act of the show ending, the second act beginning, and the camera fading back in. It is important to note that the action sequence “INT. JAKE'S HOUSE (DISTANT FUTURE)” is not a structural element, because it is telling the reader that the current scene is set at Jake’s house. This is equivalent to the author of a novel saying where the next scene occurs, which is an essential detail of the narrative, and not structural information. Following the location of the new scene, is a description of what is happening: Old Jake is sitting, and Melanie is watching him. Finally, the dialogue of the scene starts.



JAKE

Dad... ?

SISKO

What... what happened... ?

But before Jake can reply, Sisko's body starts to FLICKER and DISSOLVE like it did in the Defiant's Engineering room...

Jake watches as the terrible moment repeats itself... until Sisko completely completely DEMATERIALIZES again…

Off Jake's confused, pained expression we...

FADE OUT.

END OF ACT ONE

DEEP SPACE NINE: "The Visitor" - REV. 08/04/95 - ACT TWO 20.

ACT TWO

FADE IN:

INT. JAKE'S HOUSE (DISTANT FUTURE)

Old Jake sits quietly, his thoughts far away in the past.

OLD JAKE

I told Dax about what'd happened...

Example 2.10: An excerpt from "Star Trek: Deep Space 9 – The Visitor" with structural elements highlighted.

Chapter 3

Annotating narrative boundaries

In 2017, I decided to work on narrative boundaries from natural text. Back then, there was no data annotated for this phenomenon. I needed a set of annotated data to teach computers to extract narrative boundaries from text, and to evaluate how accurate the computational models were. Therefore, I wrote an annotation guide, recruited annotators, trained them to annotate a text for narrative boundaries according to the rules in my annotation guide, and adjudicated their markings into a gold-standard corpus. This chapter is about my annotation study, and about a shared task I participated in regarding the annotation of narrative levels from text.

First, I will cover the logistics of the narrative boundary annotation study in §3.1. Then, in §3.2 I will talk about how inter-annotator agreement was measured. Next, I will discuss the results of the annotation study in §3.3. Finally, I will talk about the 2018 full name of SANTA before abbreviation (SANTA) workshop, in §3.4. The SANTA workshop in Hamburg, Germany, was about the construction of narrative level annotation guides. I lead one of the eight teams that participated, which involved submitting my annotation guide to be used for the workshop shared task, and doing a set of annotations according to my guide, as well as doing a set of one of the other team's guides.

3.1 Annotation study logistics

I hired two graduate students with interest and expertise in narrative to be annotators in the narrative boundary annotation study. The annotator experts recorded their judgments on a corpus of 27 texts, making up 287,777 words from various sources. The corpus is comprised of; 91,245 words from the short stories of Haruki Murakami; 85,927 words from novel excerpts; 54,293 words from TV show screenplays; and 56,312 words from court transcripts. The annotations were recorded by highlighting spans of text in PDF files that represent each narrative level. The specifics of the annotation procedure are discussed in §3.1.2.

3.1.1 Annotators and training

During the study, both of my annotators were graduate students: one studying creative writing, and the other linguistics. I chose an annotator from the creative writing department because I thought it would be useful to have annotator who has already invested significant time thinking about narrative structures, and the structure's function in telling stories. Having an annotator from creative writing was a great asset, not just for precise annotations, but for hearing specific explanations as to why they made particular choices while annotating. My conversations with the annotator positively influenced the feature design of the automatic boundary extractor.

My second annotator was from the linguistics department. I was interested in working with an annotator who was a linguist, because I thought it would be interesting to observe how someone studying the way words are used to communicate meaning would interpret narrative boundaries. It was helpful to speak to an annotator about the subtleties of where the narrative boundaries occur in text.

Once recruited, I began training these two graduate students to annotate narrative boundaries on English literature. The first step of training was instructing the annotators to read the annotation guide that I wrote. My annotation guide, which is the first guide of this type, is similar in content to Chapter 2 of my dissertation. My annotation guide covers the narrative theory necessary for annotating narrative boundaries. The guide also gives examples from novels and TV show transcripts, which illustrate the occurrence of different types of narrative boundaries, and how to make annotation decisions in difficult situations.

The second step of training was discussing the annotation guide as a group. Meeting both annotators in person was useful, because I could clarify disagreeing annotations, and discuss confusing situations. Then I showed the annotators how to record their findings using the highlight feature in any PDF viewer. I annotated some short stories by Murakami, as a group, to gain experience annotating narrative text before the official study began. After this, the annotators were ready to begin annotating for the study. To obtain two independent annotations, the annotators worked on their own; they could not communicate or ask each other questions before adjudications.

The first set of texts the annotators worked on was ten short stories from the Japanese novelist Haruki Murakami. I chose to analyze these texts because the author's use of narrative levels is especially complex and layered. Michael Seats, a Murakami scholar, said that his writing is unique because of its "… use of the 'fragment' as the minimal unit of narrative discourse…" (Seats, 2006, pp. 12). Seats argues that Murakami does all he can to subvert the “literary paradigm”, where the narrator “records” history, and the reader “witnesses” it. Instead, Murakami “…posits a transcendental narrative perspective which liberates the range of subject positions available to the reader” (Seats, 2006, pp. 14).

At first the annotators worked on annotating one text a week. I wanted to go over the annotations with the annotators weekly, so that I could monitor the agreement of the team: I wanted to catch recurrent mistakes early on, so that these common mistakes wouldn’t persist throughout the study. A common mistake at the beginning was highlighting trailing text after a narrative embedded in dialogue, as discussed in §2.2.1. Once I was satisfied with the annotators’ performance, I started assigning the annotators multiple texts at a time. They continued to read and annotate the assigned texts independently.

After assigning 10 Murakami short stories, I began to assign excerpts from novels. First, I assigned the first chapter of a novel that is known for its embedded narrative, “Heart of Darkness” by Joseph Conrad (Conrad, 2016). For the remainder of the novel excerpts, my goal was to compile a set of texts from contemporary authors of a diverse background. I chose the first two chapters from the feminist author Miranda July’s “The First Bad Man” (July, 2015).

I also choose a couple chapters from J.K. Rowling’s “Harry Potter and the Goblet of Fire” because she is one of the most popular writers of the 21st century (Rowling, 2000). Ironically, even though “Harry Potter” novels are children’s books, the topology of the narrative levels of the books are not as simple as I had expected. I thought that maybe this series could serve as a control, since it *should* have a simple or flat arrangement of narrative levels. However, I was wrong, since J.K. Rowling wrote these children’s books with a structure that was just as complex as any of the other texts in my study. If children can read and understand narrative boundaries, then computers must understand this narrative structure too. This finding has helped me realize that the use of narrative levels is an essential element of communication and storytelling.

Additionally, I chose excerpts from two novels by the Canadian author Margaret Atwood. I chose her work because her narratives and their structure are unique to her writing. In “The Blind Assassin” there are three different narrative frames: 1) newspaper clippings, 2) a first-person narrative by Iris, and 3) the pages of the fictional book *The Blind Assassin,* which is a book written by an author in the second narrative frame (Atwood, 2000). In the third narrative frame, two unnamed characters are having a conversation, where they are telling the story of a made up dystopian world, the world of the “Blind Assassin”. Narratologist Barbara Dancygier analyzes the narrative levels, or “spaces” as she labels them, in her book “The Language of Stories: A Cognitive Approach” (Dancyiger, 2012). Regarding “The Blind Assassin” and other narratives, she says “The text is the form, while the story is what the text represents. While the text may be fragmented, incoherent, or temporally disorganized, the story is a temporal sequence of causally linked events leading up to a resolution of some conflict or problem” (Dancyiger, 2012, p. 53). I also chose “The Handmaid’s Tale” (Atwood, 2017) because of the interlaced narrative structure: the narrator frequently tells stories from her past in the narrative speech, and hears fragments of other people’s stories in dialogue. The narrator even has psychological *ticks* where she repeats messages that were instructed to her by Aunt Lydia.

Once the annotators finished analyzing and annotating the novels, I had them work on TV show transcripts. Before the annotators recorded their annotations, I instructed them to watch the episodes of the TV shows. Then they were to read through the scripts, and highlight the correct narrative levels. They annotated an episode of “Seinfeld” called “The Bris” (Charles, 1993). I selected this episode because there are so many narratives embedded in the dialogue, and multiple overlaps between narrative levels.

Next, I had them annotate the “Bad Blood” episode of “The X-Files” (Gilligan, 1998) because it has three embedded narratives that tell the same story, but from three different perspectives. The last two TV shows were two episodes of “Star Trek: The Next Generation” (Moore, 1993 & 1994). These are episodes with lots of time travel, tellings of past and future stories, visions, and dreams. Hence, they made for compelling texts for narrative level annotation. The final three TV shows selected were from the science fiction genre, not because of a conscious decision on my part, but because they had atypical story telling methods.

Finally, I had the annotators work on transcripts of court cases. These cases included excerpts of Monica Lewinsky’s grand jury testimony (2000), and excerpts from two supreme court cases: Obergefell v. Hodges (2015), and Bush v. Gore (2000). I chose court cases because they are rich with embedded narratives about the past, and I also wanted to see what would happen when I analyzed the narrative boundaries of court proceedings. This work might prove useful in advancing computational understanding of court cases.

3.1.2 Annotation procedure

For the novels, short stories, and court cases, I instructed each annotator read through the full text once, without making any annotations. Then, they reread the text, this time highlighting the spans of each narrative level in a unique PDF. For each unique narrative level, the annotators were instructed to make a copy of the PDF. Then they used the highlight feature of a PDF viewer, and highlighted the spans of text the current narrative level occupies.

Then, I had the annotators fill out a spreadsheet with some metadata for each narrative level, including details such as what is the POV of the narrator for each narrative, if the current narrative is embedded then what narrative is it embedded in, and whether the narrative contains a time shift, vision, or dream. These extra bits of metadata were included because I wanted the annotator to be consciously aware of these characteristics while recording their decisions. A final data point was collected: a title for each narrative level. This was done so that the annotators could have a name with semantic meaning for each narrative, and not just a number to represent it.

3.2 Agreement metrics

I used two agreement metrics to measure inter-annotator agreement: Cohen's Kappa (Koch, 1977), and a metric I devised, *agreement relative to disagreement* or ARD. Both metrics are first calculated for each narrative level. Then the agreement for each narrative level from a text are averaged together. When I report an agreement for a text, as in Table

3.1 – 3.4, note that I am reporting the average agreement, for that metric, averaged over the agreement for each narrative level.

	annotator 1	
annotator 2	true positives	false positives
	false negatives	true negatives

Figure 3.1 Confusion matrix

3.2.1 Cohen's Kappa for span agreement

A value for Cohen's Kappa (Koch, 1977) is calculated for each narrative level. The ideal value of Cohen's Kappa is 1, which denotes perfect agreement. The range for Cohen's Kappa is between -1 and 1. Any value of Kappa below 0 is considered to have no agreement, between 0 and 0.2 slight agreement, between 0.2 and 0.4 fair agreement, between 0.4 and 0.6 moderate agreement, between 0.6 and 0.8 substantial agreement, and 0.8 and 1 is almost perfect agreement (Koch, 1977).

To calculate Cohen's Kappa, I populate a confusion matrix (Figure 3.1) for each narrative level. With respect to a narrative level, this is how the confusion matrix is populated:

True positives: the length in characters of the text spans that both annotators highlighted for a narrative level.

True negatives: the length in characters of the text spans that both annotators did not highlight for a narrative level. This measures how long are the spans of text that both annotators said do not belong to a narrative level.

False positives: the length in characters of text spans where annotator 2 said belong to a specific narrative level, while annotator 1 said it did not belong to this level. This is a disagreement.

False negatives: the length in characters of text spans where annotator 1 said belong to a specific narrative level, while annotator 2 said it did not belong to this level. This is also disagreement.

The values in each quadrant of the confusion matrix are the sums of the lengths of a span meeting a certain condition. The lengths of each span are calculated by counting the number of text characters in the span.

3.2.2 ARD for span agreement

A value for agreement relative to disagreement is calculated for each narrative level. Again, for each narrative level the confusion matrix is populated using the same procedure as in §3.2.1. Then the value of the ARD metric is calculated for each narrative level using Equation 3.1. Finally, to obtain the ARD for a full text, the ARD of each narrative level from the text is averaged together.

The goal of this metric is to measure how long the spans of annotator disagreement are relative to the length of agreement for inclusion in the level. To calculate this, I add up the length of the spans where the annotators disagree. This is the sum of the false positives and false negatives in the confusion matrix. Then, I normalize the sum by the length of the spans where the annotators agree, represented by the value for true positives in the confusion matrix. Right now, this ratio gives us disagreement with respect to agreement. To covert this into a number that represents agreement (and not disagreement) I subtract the disagreement ratio from the number one.

$$ARD = 1 - \left(\frac{false\ positives + false\ negatives}{true\ positives} \right)$$

Equation 3.1 Equation for agreement relative to disagreement (ARD)

The range of values for ARD is from negative infinity to positive one. Perfect agreement is when ARD is positive one. Anything below zero indicates less agreement that disagreement for the current narrative level. Our goal is to have high agreement, and this is represented by an ARD close to one.

Before I explain the results, let's discuss the difference between Cohen's Kappa and ARD. For the Murakami short stories, the average Cohen's Kappa was 0.94, and the average ARD was 0.83. For the corpus of novels, the average Cohen's Kappa was 0.93 and the average ARD was 0.81. In these experiments the ARD is a harder metric than Cohen's Kappa.

3.3 Results

Corpus	Cohen's Kappa	ARD	Sentences	Words	Narrative levels
Court case transcripts	0.78	0.66	2870	56312	42
Novels	0.93	0.8	5099	85927	124
Murakami's short stories	0.94	0.83	6642	91245	101
TV show transcripts	0.86	0.74	4472	54293	57
		Total:	19083	28777	324

Table 3.1 Summary of inter-annotator agreements

In this section I present the results for the four types of texts. Table 3.1 summarizes the results by type of text. Here the Cohen's Kappa and ARD columns represent the average inter-annotator agreements for all the texts of the same type. The sentence column represents how many sentences for that type of text were used in my study. The words column represents how many words for that type of text were included in my study.

3.3.1 Short stories by Haruki Murakami

The inter-annotator agreements for the Murakami short stories are the highest of the text types in the study, across both agreement metrics. It is possible the results are the best since I trained my annotators using other short stories by Murakami that were not used for creating the gold-standard. Typically, inter-annotator agreement gradually rises as an annotation study progresses, but in our study, the annotators first worked on the ten Murakami texts.

I think the high agreement is due to how Murakami uses narrative levels as a literary device in his writing. Changing narrative level is essential for the stories Murakami tells. The reader cannot passively consume these short stories and not be aware of the ever-changing narrative levels. Changes in these narrative levels are front and center in these stories, which is why I think the annotators were so successful in identifying them.

Title	Cohen's Kappa	ARD	Sentences	Words	Narrative levels
All God's Children Can Dance	0.92	0.78	470	7642	11
Birthday Girl	0.95	0.91	401	5894	4
Drive My Car	0.96	0.94	1000	13166	6
Honey Pie	0.94	0.85	831	11353	16
Landscape with Flatiron	0.87	0.69	572	7184	6
Scheherazade	0.94	0.71	728	10837	13
Super Frog Saves Tokyo	0.87	0.61	610	8214	13
Thailand	0.95	0.89	510	7342	8
UFO in Kushiro	0.97	0.93	534	6877	3
Yesterday	0.98	0.96	986	12736	21
Average:	0.94	0.83			
		Total:	6642	91245	

Table 3.2 Short story inter-annotator agreement results

3.3.2 Excerpts of novels

The inter-annotator agreement for excerpts from novels was the second highest of all the text types, on both metrics. The results are on par with the agreements for short stories.

This is most likely because both sets of texts are written by professional/ lauded authors, who are writing fiction, that is intended to be read by readers. These texts are *designed* to be consumed by readers, and the authors want the readers to be able recognize, and keep track of each narrative level, so that they can make sense of the story.

Author	Title	Cohen's Kappa	ARD	Sentences	Words	Narrative levels
Margaret Atwood	The Blind Assassin	0.96	0.91	778	10632	24
Margaret Atwood	The Handmaid's Tale	0.93	0.83	799	13359	21
Joseph Conrad	Heart of Darkness	0.97	0.99	887	17217	2
Miranda July	The First Bad Man, chapter 1	0.93	0.77	234	3043	7
Miranda July	The First Bad Man, chapter 2	0.87	0.47	568	7856	12
James McBride	The Good Luck Bird	0.94	0.84	591	10876	27
Claudia Rankine	Citizen	0.88	0.69	669	12017	19
J.K. Rowling	Harry Potter and the Goblet of Fire Chapter 2 and 3	0.94	0.87	573	10927	12
	Average:	0.93	0.8			
			Total:	5099	85927	

Table 3.3 Novel inter-annotator agreement results

3.3.3 TV show transcripts

Show	Title	Cohen's Kappa	ARD	Sentences	Words	Narrative levels
Seinfeld	The Bris	0.94	0.86	644	5908	5
Star Trek: The Next Generation	Tapestry	0.84	0.81	986	12683	17
Star Trek: The Next Generation	All Good Things…	0.81	0.59	2172	26496	22
The X-Files	Bad Blood	0.85	0.7	670	9206	13
	Average:	0.86	0.74			
			Total:	4472	54293	

Table 3.4 TV show transcript inter-annotator agreement results

The inter-annotator agreements for TV show scripts are a bit lower than those of short stories and novels. The agreement is still high, but it's not as agreeable. The concept of narrative level was created to explain literature. Using the narrative level construct to analyze TV show scripts is not as straightforward as it is for literature. There are some difficulties: structural text like action, scene changes, and character speech markers are recurring elements of scripts, and these elements are discussed in §2.3. Character speech markers are like a novel announcing who is speaking (i.e. "Harry Potter said…"). Many of the annotation disagreements for TV show scripts came from being unsure how to handle the structural text. The agreement was still above 0.80 on the Cohen's Kappa metric, which is near perfect agreement, but it was still harder to annotate than the texts from literature.

3.3.4 Court case transcripts

Title	Cohen's Kappa	ARD	Sentences	Words	Narrative levels
Bush v. Gore	0.69	0.54	468	12126	5
Comey testimony	0.79	0.68	540	11888	11
Lewinsky testimony	0.83	0.71	996	12669	10
Obergefell v. Hodges	0.81	0.7	866	19629	16
Average:	0.78	0.66			
		Total:	2870	56312	

Table 3.5 Court case transcript inter-annotator agreement results

It is interesting that the lowest agreement scores are from the court cases. These are the only texts that weren't designed by writers, for readers or viewers to consume their stories. The conversations were recorded by court transcripts. They are spontaneous, and the speakers don't have time to edit and distill their thoughts into clear narratives. On the other hand, writers of novels and short stories take time to plan, write, edit their stories, and think about how they will present their narratives to be read.

Therefore, it should be expected that the court cases have the lowest agreement for narrative level identification. These texts are rawer, and noisier than any other texts in the study. The speakers interrupt each other, they deflect questions, and not all the text is narrative discourse. Despite these realities, I still chose to include court case transcripts in the study, because computers need to be able to understand noisy and complicated texts as well as understand noisy and complicated people. This was a first effort in annotating narrative levels in court transcripts, and the agreement was substantial, but there is room for improvement. I think it would be useful to spend more time annotating court case transcripts as a group before assigning new texts to the annotators.

3.4 SANTA Workshop shared task

After I completed my annotation study I participated in a shared task on annotating narrative levels. Usually, shared tasks are competitions where teams of NLP researchers submit computational models that are trained to solve a specific problem. Shared tasks "… are indisputably drivers of progress and interest for problems in NLP. … Shared tasks revolve around two aspects: research advancement and competition" (Nasim, 2017). Each model is evaluated on a set of test problems, the set of models are ranked by their performance, and a winner is usually chosen. In the case of SANTA, the first phase focused on comparing narrative level annotation guides, and the second phase will evaluate the performance of different programs that can automatically extract narrative levels from English literature. At the time of writing, phase 1 is currently underway, and phase 2 will start in 2020.

The first phase of the SANTA shared task occurred just a few months after I finished creating my gold-standard. The task began in 2018, when a group in Germany organized a shared annotation task on *narrative levels*. The shared task was called Systematic Analysis of Narrative Texts through Annotation or SANTA.

I participated, along with teams from seven other universities. Each of the eight teams submitted an annotation guide, annotated a set of eight short texts according to their own annotation guide, and annotated the same texts according to one of the other groups guides. Also, a team of students at the University of Hamburg was trained to annotate according to each guideline, and carried out a third set of annotations according to each annotation guide. This was done to compare the effectiveness of each annotation guide. The shared annotation task is only the first phase of the SANTA task. Phase 2 is a competition on the automatic extraction of narrative levels on natural text. This phase will be held in 2020, and I will compete using the programs based on those discussed in Chapter 4. I plan to improve the narrative level extractor over the next year.

Back in 2017 when there were no texts annotated for narrative levels. My only option was to create my own corpus. However, there is finally interest in this task, and there is a community that is now trying to figure out affective methods for the annotation of the narrative level phenomena, but before recently there were no resources available. At the time of writing, my work is the only publicly shared work detailing experiments on the automatic extraction of narrative boundaries. In 2020, the second phase of SANTA will begin, and other researchers will have their narrative level extractors compete against mine to see which one is the most accurate. At the time of writing, the study of the computational understanding of narrative boundaries is in a period of growth.

Some of the other teams misinterpreted my annotation guide. They were only instructed to differentiate between embedded and interruptive narratives, but they thought that they needed to say whether each narrative level was a flashback, flashforward, vision, or dream. This was an incorrect assumption, and nowhere in our annotation guide did it say that this was something they were supposed to annotate. However, my annotation guide gives examples about how time shifts, and dreams or visions, can be used to signal certain changes in narrative levels. This confusion was not a problem for the annotation study that I ran. However, for SANTA phase 1, I was not able to train the other annotators to use our annotation guide. They just read the guide, and annotated without having a chance to speak to me about how the guide should be applied.

Ideally a guide should be enough to teach an annotator to make annotations, but in practice it is much better to personally train annotators how to make annotations. Of course, an annotation guide is necessary for establishing a theoretical common ground, and rules for making decisions. Still, there can be a gap when trying to go from theory to making annotation decisions. I have found that doing annotations with your annotators, and having open discussions about why annotation decisions are being made is invaluable for harvesting quality annotations.

Early on in the workshop, there seemed to be two reasons people wanted to produce annotation guides for narrative levels: 1) researchers who wanted to use these guides, to produce annotations with high inter-annotator agreement to use for training and testing of computational models and 2) narratologists or digital humanities scholars who wanted to create annotation guides, each with different interpretations of the concept of narrative level, which can be used to fuel discussion about narrative levels. Most of the teams could be grouped into the narratologists camp. Personally, I participated because of my interest in using annotations for supervised machine learning. Other than myself, there was only one other team who participated because they were interested in using these annotations for construction computational models of narrative levels.

This made for an interesting workshop, because my goal was the inverse of the narratologists' goal. The narratologists were interested in producing guides that use different theories of narrative levels, so that when texts are annotated according to the guidelines, the annotations would be different. The narratologists were more interested in getting annotations with different markings for each narrative level, because they wanted to discuss the differences in the schools of thought for narrative theory.

This is at odds to my goal, because I want a set of annotations, which represent the narrative boundaries, and to treat these annotations as canonical. I went into the workshop thinking everyone wanted to harvest annotations with high inter-annotator agreement. I had this assumption because the second phase of the workshop is a competition on the automatic extraction of narrative boundaries from text. I thought that since this is the goal of the second phase of the workshop, that the goal of the first phase would be to construct an annotation guide, or an amalgamation of annotation guides that would enable annotators to produce accurate annotations.

Chapter 4

Automatic extraction of narrative levels from text

The main accomplishment of my dissertation is the identification of features that allow for the automatic extraction of narrative levels from long form texts. Specifically, identifying point of view, diegesis, story content, and when main characters are mentioned in order to automatically extract narrative levels.

Once again, narrative boundaries are the locations in text between different narrative levels. Narrative levels are the spans of text where different narratives are being told. Each narrative level has at least two narrative boundaries, the beginning and end of that narrative, but often a narrative can occupy different non-contiguous spans of text. A narrative is a telling of a story. A story is a series of events that is narrated over time. Sometimes an event, that takes place in the world of a story, results in a new narrative. For example, this can occur in dialogue when a character may tell a story to another character. This is an example of an embedded narrative. On the other hand, the telling of the current story could suddenly be interrupted, and a new narrative takes over. The interruptive narrative level phenomena are less common than embedded narratives, but it is still an element of narrative structure. Please refer to §2.2 for a more rigorous definition of narrative levels, and how they are used in storytelling.

Why is it important to detect the location of narrative boundaries in text? Higher level information extraction, like plot extraction, and event coreference, would be inaccurate and nonsensical if narrative levels have not been distinguished. How could a computer decide what the plot of a story is, if it does not know what span of text the story is told in? Narrative level extraction is a necessary first step for parsing narrative text; It reveals the structure of which narratives occupy which spans of text, and then finer grained analysis can be carried out on each distinct level.

In this chapter, I will briefly look at the concept of narrative boundary, and other narratological characteristics that are necessary for this discussion. For a more in-depth discussion of the theory, please see Chapter 2. In §4.1, I will discuss the pipeline for extracting features used for automatic narrative level extraction. In §4.2 I will discuss the experiments used to evaluate the performance of the automated extractions. In §4.3 I will discuss the findings.

I plan to submit the contents of this chapter, and the previous chapter, to one of the main Association of Computational Linguistics conferences (ACL, EMNLP, NAACL, or EACL).[4]

4.1 Designing the narrative level extractor

4.1.1 Discussion of the scope of the work

In order to design this project, I examined the scope of the problem: what granularity of text spans should the computer make decisions about, and what options should the computer decide between?

[4] https://www.aclweb.org/portal/

For the experiments in this chapter, I decided that given a long narrative text as an input, the computer must decide which sentences belong to an embedded narrative. This is framing the problem of detecting narrative levels as binary classification, where the computer decides whether each sentence belongs to the original, lowest level of narrative in a text, or not (i.e. the sentence is part of an embedded or interruptive level). This is a simplification of the problem of narrative level extraction. Next, I will discuss the simplifications, and talk about what is lost when I make these generalizations.

First, narrative boundaries might occur at any point in a sentence, but the programs discussed in this chapter make classifications about entire sentences. The computer is deciding if a sentence contains text from different levels of narrative or not. With respect to the gold-standard annotations, a sentence contains text from a different level of narrative if there is an embedded or interruptive narrative anywhere in that sentence. Ideally, the classifier should be able to look within a sentence, and figure out between which words a narrative boundary occurs. At this stage of research, I am trying to figure out if there is a change of narrative level within a sentence.

Second, I treat interruptive narratives like embedded narratives because of our scope. Although interruptive narratives are theoretically different than embedded narratives, they both represent a change of the story being told. The main difference is that for embedded narratives, there is an event in the lower level that leads to a telling of a new narrative, while in an interruptive narrative there isn't necessarily an event that causes the new narrative to be told—the new narrative interrupts the telling of the old narrative.

Third, the detector can only decide if there is a new narrative level or not. In reality, this is more complicated than a binary decision. There can be many levels of embedding, in some of our texts there were three levels of embedding (a narrative, within a narrative, within a narrative). It is important to detect whether a narrative is embedded or interruptive. It is possible though to rerun the extractor on text from an embedded narrative, and decide recursively whether there is another level of embedding.

Limiting the scope of the task enabled me to design a narrative level extractor that can do a very specific task: decide whether there is a change in narrative levels. I have not run experiments about what type of boundary is between the levels, or how many levels of embedding is occurring. To be sure, this is a simplification of the depth of the full problem of narrative level parsing/extraction, but my work shows a positive result, which is the first result published for the task of narrative boundary extraction. Although I am not attempting to solve every part of the problem, my work can be used to help make the more complicated decisions and to point toward what types of features should be used. My work is an important first step pointing the way toward finer grained narrative level extraction.

4.2 Feature extraction

In this section, I will discuss the design of the programs I used to automatically extract features for narrative levels from raw text. All programs described in this chapter were implemented in Java 8 (Gosling, 2014), using the Eclipse IDE[5]. There are two main types of programs that I implemented for this work: 1) programs that extract the value of features from raw text, and save them to a cache/disk, and 2) programs that use the values of specific

[5] https://www.eclipse.org/ide/

features from specific texts, to train a computation model, and evaluate the performance of this model. The two phases of programs can be visualized in Figure 4.1.

I used a two-phase approach to this problem because feature extraction can take a long time. Some of the features, like story-content and diegesis, can take an hour to extract for the texts in this study. Saving the values of the features for each text to a disk allowed me to quickly train support vector machine (SVM) models, and run many experiments using different sets of features. In prior projects, especially with the story extractor (Chapter 5), I was not able to run as many experiments on my SVM models, because each time I trained a model, I would need to re-extract the same features from raw text. I was wasting time—computationally—extracting the same features from text every time I needed to train or test a SVM model.

After wasting so much time waiting for the same features to get extracted for my story extractor experiments, I made a design decision for the narrative level experiments:

1) Extract the features once, and save them to disk
2) Load the features to vectors when training or testing SVM models

It might seem that this is an obvious design choice, but it is something that I didn't have the foresight to implement until after I had already wasted countless hours waiting for the same features to get extracted for my story extractor. Having all the features cached allowed me to run experiments where I could tweak the encodings of the features or the values of the SVM hyperparameters. Although it took longer to write code that serialized the feature values to disk, it saved time in the long-run, because the features only had to be extracted once.

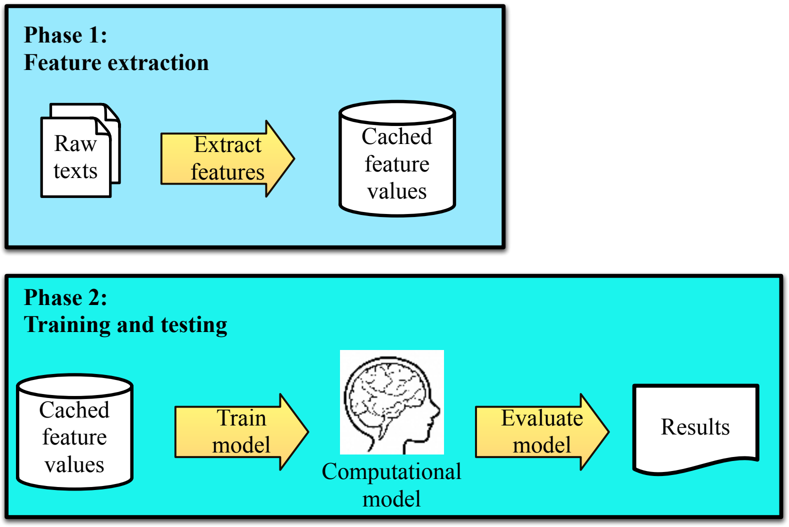


Figure 4.1 The two phases of programs for narrative level extractor

Before I cover the details of how the features were extracted, let’s list the types of features that were extracted and experimented on:

- First, there were some features that represent narrative characteristics, like the point of view of the narrator, narrative diegesis and story content. The first two features give information about how the narrator is telling their narrative. Changes in these characteristics of narrators can signal changes in narrative level. Story content classification gives the computer awareness of how story-like a paragraph is, which is useful for determining if a span of text is embedded.

- Second, character-based features were extracted, because knowledge of which characters appear when, and whether they are *main characters* is important for classifying narrative levels.
- Third, features based on word-embeddings (Mikolov, 2013). Unfortunately, these features lead to the training of models which were unable to extract narrative boundaries as well as the features from the first three classes of this list. I still discuss the engineering and the failure of these features to classify narrative levels, to provide transparency on a negative result.

4.2.1 Extraction of narrative characteristics features

Let's first discuss the feature extractors that use pipelines developed in my prior work. In this chapter, I will treat the POV, diegesis, and story classifiers as *black boxes* which produce binary classifications. In reality, these individual boxes are complicated processes, each with their own unique feature extraction pipelines. See §5.3 for story classification, §6.4.3 for POV extraction, and to §6.4.4 for diegesis classification. See Figure 4.2 for the pipeline that extracts POV, diegesis, and story classifications from raw texts, and caches the feature values for use in the training and testing of computational models in phase 2.

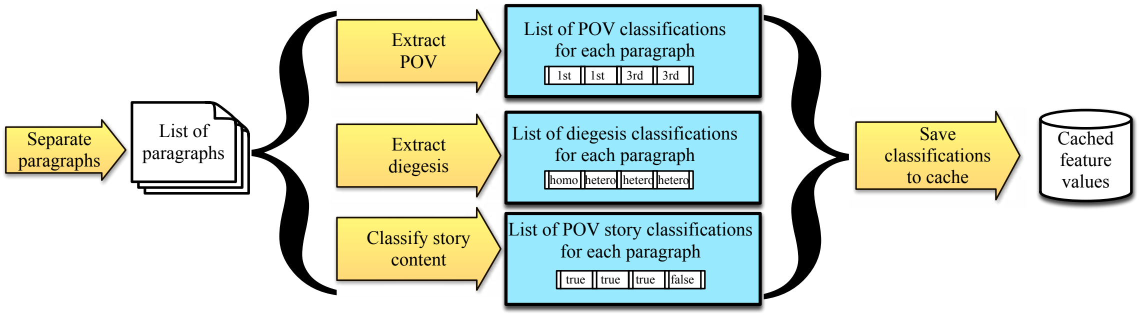

Separate paragraphs
List of paragraphs
Extract POV
List of POV classifications for each paragraph
1st
1st
3rd
3rd
Extract diegesis
List of diegesis classifications for each paragraph
homo
hetero
hetero
hetero
Classify story content
List of POV story classifications for each paragraph
true
true
true
false
Save classifications to cache
Cached feature values

Figure 4.2 Pipeline for narrative characteristics feature extraction

When extracting features for narrative level detection, the POV, diegesis, and story classifiers were run on each paragraph of text as inputs. This contrasts with running the classifiers on texts of sentence granularity. This is important to note, because the annotations are on the sentence granularity, and narrative levels can occur within a paragraph. However, the story extractor was trained on paragraphs of text, and the POV and diegesis classifiers were trained on texts that are around a page long, or a few paragraphs.

Although it would be ideal to run the narrative characteristic classifiers on each sentence of the text, the implementations are inaccurate when run on short texts, like single sentences. I decided it would be better to give the narrative characteristic classifiers texts that were closer to the length of which they were trained. So, I ran them on texts that were a paragraph long. Going forward, it might be beneficial to retain the narrative characteristics classifiers on texts of varying sizes, especially for short spans of texts.

Also, I made a slight adjustment to the POV and diegesis classifiers: Usually the classifiers do not extract features from text that is within quotation marks, because this is usually not part of the narrator's narrative. But, for the process of identifying narrative boundaries, I want to know how the POV and diegesis changes in the text that is within quotes. To rectify this, I modified the POV and diegesis classifier, so that the text in quotes is not removed. Hence, all text from each paragraph is analyzed by the POV and diegesis classifiers, including quoted text. It would be a great loss, with respect to finding narratives embedded in dialogue, if the POV and diegesis detectors could not analyze quoted text.

Ideally, I want to generate classifications for each sentence, but due to the design of my POV, diegesis, and story classifiers I can only generate classifications for each full paragraph. I want sentence level classifications because the SVM for narrative level extraction uses feature vectors from each sentence. The following is the procedure for propagating paragraph level classifications (like POV, diegesis, and story content) down to the sentence level:

1) A raw text is broken up into a list of paragraphs, and each paragraph is broken up into a list of sentences.
2) Each paragraph is classified for all three narrative characteristics (POV, diegesis, and story content).
3) For each sentence, the narrative characteristic classification of the paragraph that the sentence belongs to is assigned to the current sentence.

For an illustration of this process, see Figure 4.3, which shows the procedure for going from a list of paragraphs, all the way down to sentence level POV classifications. This propagation process is done three times for each text, once for each of the three narrative characteristics.

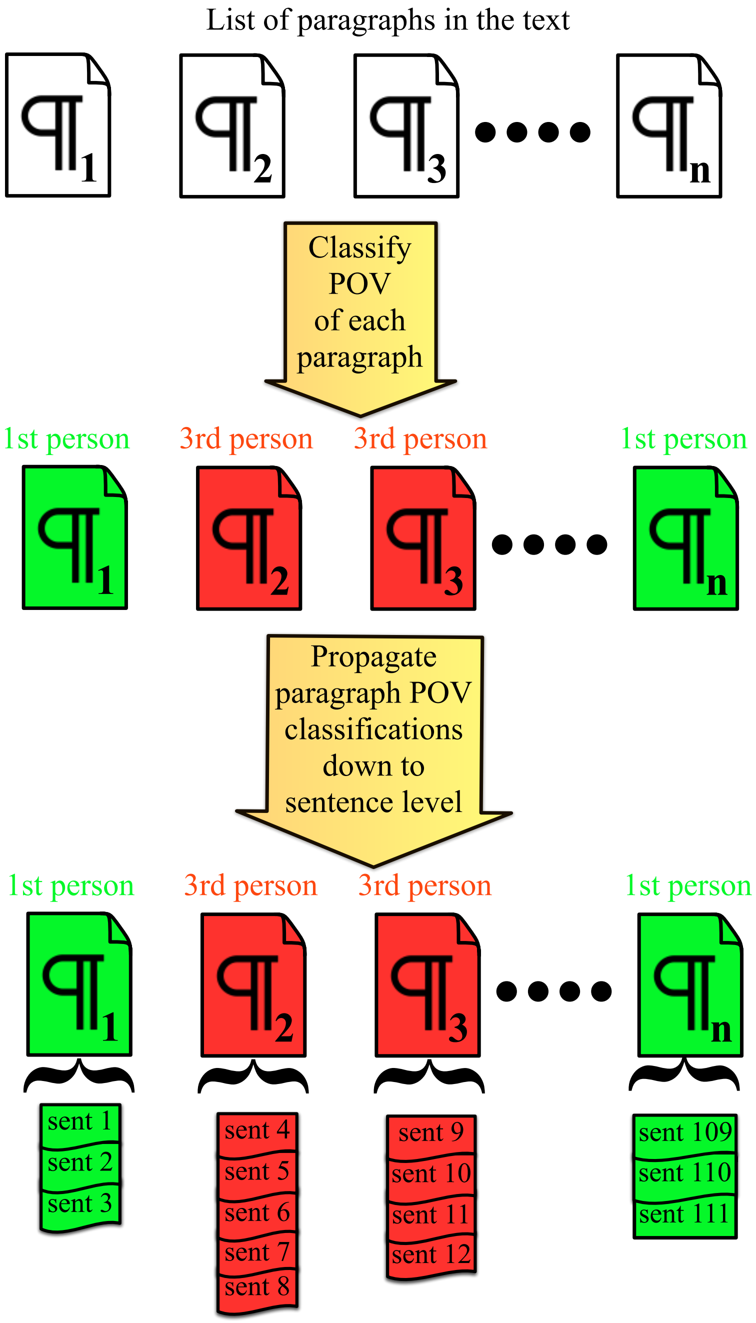


Figure 4.3 Procedure for paragraph to sentence propagation

It is useful to note, that on a 2014 MacBook Pro, with 16 GB of RAM, and a 2.6 GHz Intel Core i7 processor, the story detector takes about 40 minutes to classify each text, which range between 7,000 and 15,000 words in length. The diegesis detector takes about 25 minutes on average. The POV detector is much quicker, and takes less than a minute to classify a full text.

4.2.2 Extraction of character features

Preprocessing raw text is usually the first process in an NLP pipeline. In my pipeline, preprocessing was necessary for extracting the character features. The preprocessing pipeline detailed in this section was used by the character feature extractors.

The character based features were created specifically for the task of automated narrative boundary extraction. Therefore, I had to implement a new pipeline for these features. A good way to simulate knowledge of which characters occur when, is to analyze coreference chains. A coreference chain is a data structure that represents the mentions of an entity in a text. I assume that each coreference chain represents a character in narration being told.

Often, automatically extracted coreferences do in fact represent characters from the narrations. The longest coreference chains tend to represent characters that are mentioned the most in a novel. For a succinct example, let’s consider “Harry Potter and the Goblet of Fire” (Rowling, 2000). The longest coreference chain, according to when I ran Stanford CoreNLP’s Coreference Chain extractor, is for “Harry Potter”. This is hardly a surprising result. Why?

The first step in extracting the character features is to use Stanford CoreNLP for preprocessing (Manning, 2014). Specifically, I use tokenization, sentence splitting, part of speech tagging, lemmatization, named entity recognition, and parse tree generation. Once this preprocessing pipeline has been executed, I use the Stanford Coreference Resolution program to extract coreference chains from the entire document (Clark, 2016). Then I process the list of coreference chains that Stanford produces, and sort them with respect to the length of each chain. This process is illustrated in Figure 4.3.

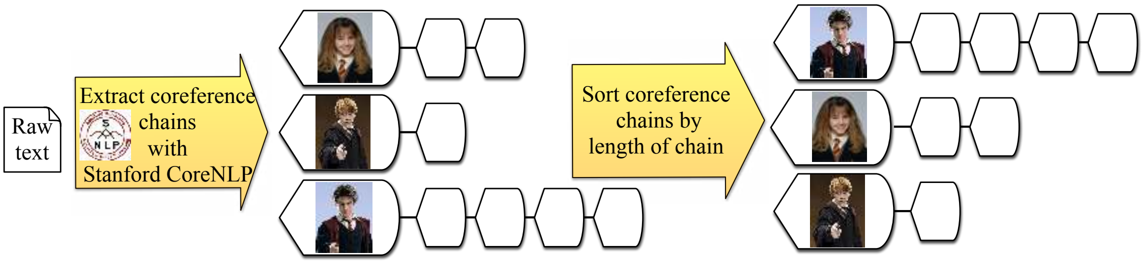

Raw text
Extract coreference chains with Stanford CoreNLP
Sort coreference chains by length of chain

Figure 4.4 Pipeline for sorting coreference chains

At this point two different sets of features were extracted:

1) The *main character feature* represents whether the main character is mentioned in each sentence. The design of this feature assumes that the main character is the referent referred to by the longest coreference chain. This feature is extracted by looking at the longest coreference chain in each text. The value of the *main character feature* is true for a sentence if the character is mentioned in that respective sentence. If the *main character* is mentioned in a sentence, then the *main character feature value* is encoded as 1, or true, to represent that the main character

is in fact mentioned. If the *main character* is not mentioned then the *main character feature value* is encoded as -1, or false, to represent that the main character is not mentioned in the current sentence.

2) The *main character sum feature* represents how many of the top three main characters are mentioned in each sentence. This feature's design also assumes that the main characters are the referents referred to in the longest coreference chains. Instead of just paying attention to the most mentioned character, I am counting how many of the three most mentioned characters appear in each sentence. The value of the feature is calculated for each sentence by looking at the three longest coreference chains, and count how many of them have a mention in the sentence of interest. There are 4 possible encodings for this feature: 0 if none of the top three main characters are mentioned, 1 if only one of the main characters are mentioned, 2 if two of the main characters are mentioned, and 3 all three of the main characters are mentioned. This feature is calculated for each sentence.

Once the values of both features, for an entire text, are extracted they are saved to disk. This is done by writing the values of each type of feature are written to a text file. The feature values for each sentence are all written on the first line of the text file, with each sentence value delimited by white space. The process of extracting the character features on a 10,000-word text takes between five and ten minutes on a 2014 MacBook Pro, with 16 GB of RAM and a 2.6 GHz Intel Core i7 processor.

4.2.3 Extraction of semantic embeddings

A third type of feature was extracted from each text: features based on word embeddings (CITE). I used DeepLearning4J[6] to load Google's 300 Word2Vec model (Mikolov, 2013)[7]. This model was trained on 100 billion words of newswire. The model is able to map from English words to vectors of length 300 that has the semantics of the word embedded into the encoding.

I used Google's pretrained model to get word embeddings for each word in my corpus. Then using vector algebra, I summed together the vectors for each word in a sentence, and normalized the length of the vector. I cached the vector that represented each sentence, so that I could experiment with using them to obtain some features. Here are three types of sentence vector based features I tested:

1) The cosine similarity between the vectors for the current sentence and the previous sentence.
2) The cosine similarity between the vectors for the current sentence and the next sentence.
3) The raw values of all 300 elements of the sentence vector, for either the current sentence, or the current sentence and either adjacent sentence.

[6] https://deeplearning4j.org/

[7] https://code.google.com/archive/p/word2vec/

Unfortunately, the features I tested based on sentence embeddings did not have a positive effect on the narrative level extraction process. The first set of sentences embedding features is supposed to simulate the semantic similarity between two sentences. My hypothesis was that the semantics of sentences near narrative boundaries would have low similarity. However, when I trained and tested the SVM using this feature, performance would suffer, and the models created were not viable. I also had a hypothesis that there might be some hidden (to human eyes) semantic signal for shifting between narrative levels. Therefore, I also tested using the raw vectors as features helped the extraction process, but, it did not.

4.3 Experimental procedure

First, features are extracted for each text, and saved to disk. As stated in 4.3 this process is separated from the SVM model training and testing to save time. The feature extraction pipeline can take about two hours to extract each feature from a single text. Extracting each feature from the full short story corpus takes around 20 hours. It would be a waste of time to re-extract the features each time a model is trained or tested.

Once features are extracted, the SVM models can be trained. In §4.3.1 I will discuss how I use cross-validation to train models and evaluate their performance. In §4.3.2 I will discuss how undersampling is used to balance the training folds evenly between the different classes. §4.3.3 lists the hyperparameters for the SVM and Java library that implements it. In §4.3.4 I discuss how I go from the gold-standard annotations to sentence annotations that can be used for training and testing the SVM models.

4.3.1 Leave-one-out cross-validation

When I trained an SVM model, and tested its performance using a variation of *leave-one-out* cross-validation (Stone, 1974). Usually, leave-one-out cross-validation has as many training sets as there are data-points. So, if there are 100 data-points, each model will be trained on 99 points, and tested on only one, and this process will be done 100 times, so that each point can get a chance to be tested.

For narrative level extraction, the granularity of being *left-out* is a full text. For example, in the Murakami short story corpus, there are ten short stories. Hence, I use the feature vectors from nine texts to train a SVM model, and test it on the one that was left out. This process is done ten times, so that each short story in the Murakami corpus can be tested on. For the case when I am doing experiments on both the Murakami corpus and the novel excerpts, there are 18 texts in total, so there are 18 models trained, each on 17 texts, so that each of the 18 texts has a chance to be tested on.

Typically, *ten-fold* cross-validation is used. This is when the data is divided into ten equal folds, nine of the folds are used for training the computational model, and the last fold is reserved to test the performance of the model. This is carried-out ten times, so that each fold gets to be the test data. Ten-fold cross-validation is used to evaluate the SVM models in chapters 5 and 6.

I choose to use leave-one-out cross-validation, since it seemed to be a more natural fit to the task at hand. In all my previous experiments (those in chapters 5 and 6) there were hundreds, or thousands of texts that needed to be classified, and each of these texts were independent of each other. However, in the narrative level extraction experiments, there were on the order of ten texts, each with hundreds of sentences to classify. Since the sentences in each text were dependent on each other, it made sense to treat each text as an individual unit, and to either train or test on the full text.

4.3.2 Undersampling

I used undersampling during training to prevent the majority class from overwhelming the classifier (Japkowicz, 2000). In the corpus of novels and short stories annotated for narrative levels, around 60% of sentences are part of embedded or interruptive narratives. Undersampling is a technique used to help supervised machine learning classifiers learn more about a class that has a significantly smaller number of examples relative to an alternative.

To implement the undersampling, the following technique was used to augment the training processes:

1) For each span of a new narrative level, the classifier is trained on the first two and last two sentences of the narrative level. These sentences represent the positive examples, or the sentences from a narrative level that is higher than the original level, either embedded or interruptive narratives.
2) The classifier is also trained on a random set of sentences that belong to the original narrative level (from sentences that do not belong to embedded or interruptive narratives). The number of randomly selected sentences is equal to the number of sentences chosen from the boundaries in the first prescription on this list.

These rules force our classifier to undersample both the embedded/interruptive class, and the original narrative classes.

Undersampling is only done on the training data. For the testing, I must evaluate the full set of annotations, not just the boundaries of the embedded/interruptive class, and not just random selections of the original narrative class.

4.3.3 SVM

I used the Java implementation of LibSVM (Chang and Lin, 2011) to train an SVM classifier

with our narrative characteristics and character based features. The hyper-parameters for the linear kernel were $\gamma = 0.5$, $\nu = 0.5$, and $c = 20$.

4.3.4 From the gold-standard to sentence annotations

I had to decide how to interpret the narrative level annotations produced by the annotation study in chapter 3. Figure 4.5 contains an example of the beginning of the narrative level annotation spreadsheet for the novel “The Handmaid’s Tale” from the gold-standard. This spread sheet is from “The Handmaid’s Tale” by Margaret Atwood. The first column contains unique identifiers for each narrative level. The second column contains the position, with respect to characters, in a text where the starting narrative boundary occurs. The third column contains the position, with respect to characters, in a text where the ending narrative boundary occurs. The fourth column contains the name annotators used for a title of the current narrative level.

This section will define how the annotation spreadsheets are transformed into a list of narrative level annotations. Each element on this list represents whether the respective sentence is original narrative level or not (i.e. an embedded or interrupted narrative level).

Narrative ID	Span Start	Span Stop	Notes
0	0	57285	
1	0	57285	June / Offred's narrative
2	1861	2509	The aunts and guns
3	4099	4150	Aunt Lydia said…
3	5122	5296	
3	16044	16132	
3	17834	18060	

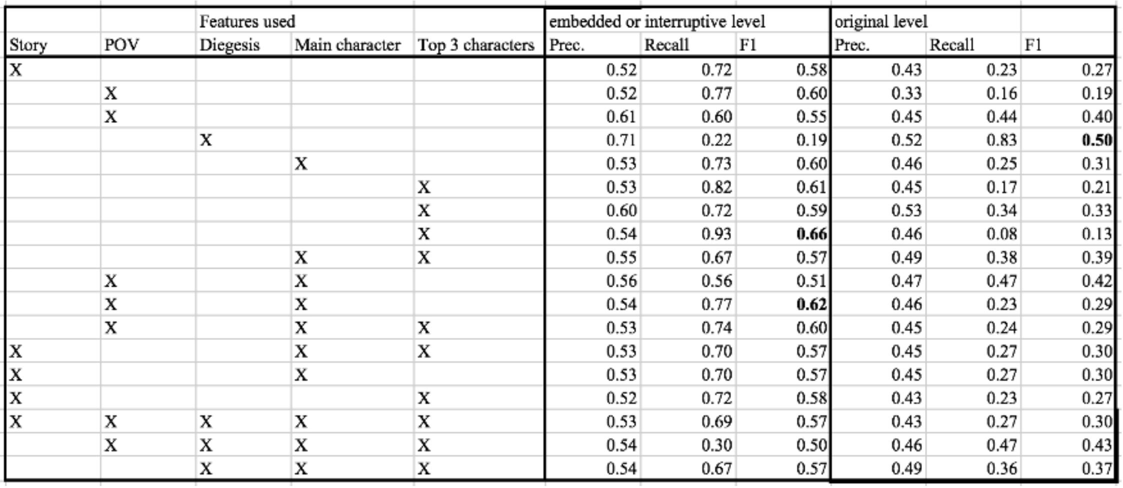
Figure 4.5 Excerpt from the gold-standard annotations.

The goal is to map the boundary locations from column two and three of Figure 4.1 into a list of binary classifications for each sentence. A classification of true means there is either an embedded or interruptive narrative in the sentence, and false means there is only the original narrative present. The following procedure was adopted: for each narrative level higher than the original narrative (the level with narrative ID 1), I look at each span's start and stop value. I map the start and stop span numbers into the sentences that the spans occur in, and make all sentences between the start and stop have annotation value for “true”, because it contains either an embedded or interruptive narrative level.

4.4 Results

In this section I will discuss different experiments run across different text corpora. That narrative based features are useful in extracting the boundaries of narrative levels in long texts. Note that the precision, recall, and F_1 metrics reported are macro-averaged over the number of folds or texts in the current corpus.

4.4.1 Short stories by Haruki Murkami

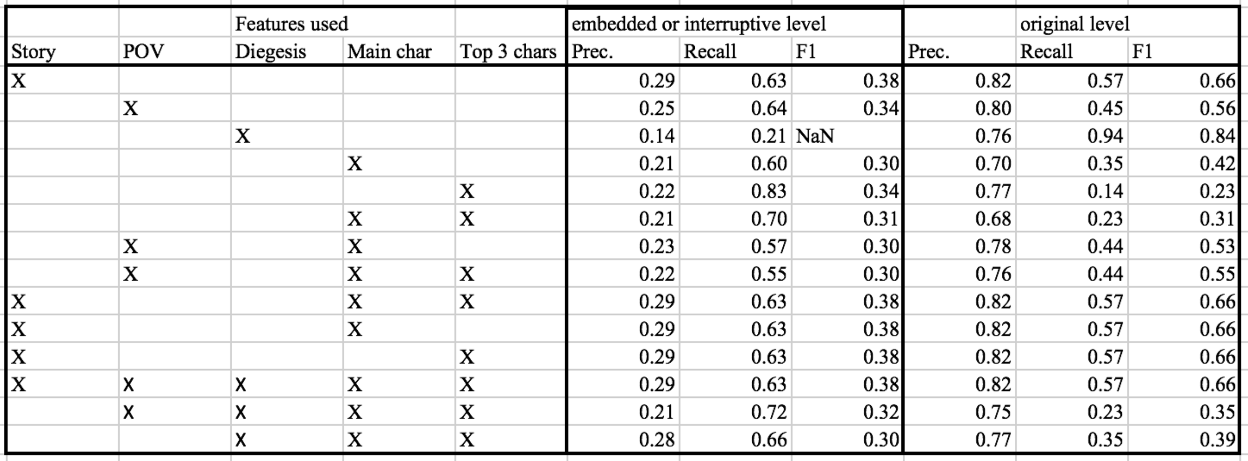


		Features used			embedded or interruptive level			original level		
Story	POV	Diegesis	Main character	Top 3 characters	Prec.	Recall	F1	Prec.	Recall	F1
X					0.52	0.72	0.58	0.43	0.23	0.27
	X				0.52	0.77	0.60	0.33	0.16	0.19
	X				0.61	0.60	0.55	0.45	0.44	0.40
		X			0.71	0.22	0.19	0.52	0.83	**0.50**
			X		0.53	0.73	0.60	0.46	0.25	0.31
				X	0.53	0.82	0.61	0.45	0.17	0.21
				X	0.60	0.72	0.59	0.53	0.34	0.33
				X	0.54	0.93	**0.66**	0.46	0.08	0.13
			X	X	0.55	0.67	0.57	0.49	0.38	0.39
	X		X		0.56	0.56	0.51	0.47	0.47	0.42
	X		X		0.54	0.77	**0.62**	0.46	0.23	0.29
	X		X	X	0.53	0.74	0.60	0.45	0.24	0.29
X			X	X	0.53	0.70	0.57	0.45	0.27	0.30
X			X		0.53	0.70	0.57	0.45	0.27	0.30
X				X	0.52	0.72	0.58	0.43	0.23	0.27
X	X	X	X	X	0.53	0.69	0.57	0.43	0.27	0.30
	X	X	X	X	0.54	0.30	0.50	0.46	0.47	0.43
		X	X	X	0.54	0.67	0.57	0.49	0.36	0.37

Table 4.1 Experiments on Haruki Murakami’s short stories

The best performing models are when the Murakami short stories are used for testing and training. The model for extracting embedded or interruptive levels has a 0.66 F_1, and it only uses a single feature, an integer that represents how many of the top three characters appear in the current sentence. The character based features do best, but the story classification feature also does well, with a 0.58 F_1.

4.4.2 Novels



		Features used			embedded or interruptive level				original level	
Story	POV	Diegesis	Main char	Top 3 chars	Prec.	Recall	F1	Prec.	Recall	F1
X					0.29	0.63	0.38	0.82	0.57	0.66
	X				0.25	0.64	0.34	0.80	0.45	0.56
		X			0.14	0.21	NaN	0.76	0.94	0.84
			X		0.21	0.60	0.30	0.70	0.35	0.42
				X	0.22	0.83	0.34	0.77	0.14	0.23
			X	X	0.21	0.70	0.31	0.68	0.23	0.31
	X		X		0.23	0.57	0.30	0.78	0.44	0.53
	X		X	X	0.22	0.55	0.30	0.76	0.44	0.55
X			X	X	0.29	0.63	0.38	0.82	0.57	0.66
X			X		0.29	0.63	0.38	0.82	0.57	0.66
X				X	0.29	0.63	0.38	0.82	0.57	0.66
X	X	X	X	X	0.29	0.63	0.38	0.82	0.57	0.66
	X	X	X	X	0.21	0.72	0.32	0.75	0.23	0.35
		X	X	X	0.28	0.66	0.30	0.77	0.35	0.39

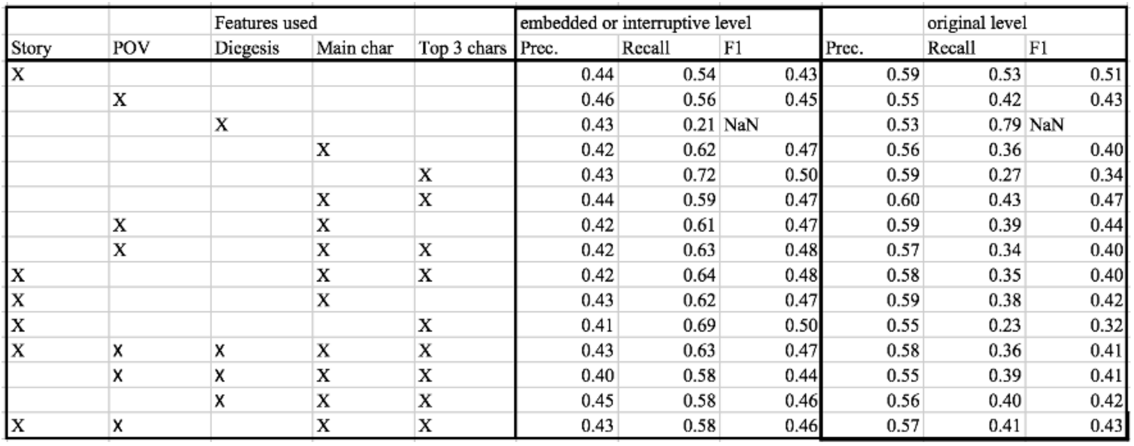
Table 4.2 Experiments on novel excerpts

The experiments on novel excerpts have better performance for the original level class than the embedded or interruptive class. However, the models trained on the short story class have better performance when classifying the embedded or interruptive classes.

4.4.3 TV show scripts and court case transcripts

Unfortunately, the features that worked with novels and short stories did not work with the TV show scripts or the court cases. When I tried training the models using these features on the script corpora, the training never converged, and the model that it produces predicts every sentence as embedded or interruptive, or it predicts that every sentence is part of the original narrative. I think this is due to the structural elements of scripts, like action, and character speech identifiers. Going forward, it might be better to parse out the action and structural elements from the dialogue.

4.4.4 Cross corpus experiments



		Features used			embedded or interruptive level				original level	
Story	POV	Diegesis	Main char	Top 3 chars	Prec.	Recall	F1	Prec.	Recall	F1
X					0.44	0.54	0.43	0.59	0.53	0.51
	X				0.46	0.56	0.45	0.55	0.42	0.43
		X			0.43	0.21	NaN	0.53	0.79	NaN
			X		0.42	0.62	0.47	0.56	0.36	0.40
				X	0.43	0.72	0.50	0.59	0.27	0.34
			X	X	0.44	0.59	0.47	0.60	0.43	0.47
	X		X		0.42	0.61	0.47	0.59	0.39	0.44
	X		X	X	0.42	0.63	0.48	0.57	0.34	0.40
X			X	X	0.42	0.64	0.48	0.58	0.35	0.40
X			X		0.43	0.62	0.47	0.59	0.38	0.42
X				X	0.41	0.69	0.50	0.55	0.23	0.32
X	x	x	X	X	0.43	0.63	0.47	0.58	0.36	0.41
	x	x	X	X	0.40	0.58	0.44	0.55	0.39	0.41
		x	X	X	0.45	0.58	0.46	0.56	0.40	0.42
X	x		X	X	0.43	0.58	0.46	0.57	0.41	0.43

Table 4.3 Experiments on the combined corpus

Table 4.3 contains the results for experiments where the models were trained on a combined corpus: a corpus with all ten of the Murakami short stories, and the eight excerpts from novels, for a total of 18 texts, and 18 folds for cross validation. With respect to the embedded or interruptive narrative level class, the results are better than the novel corpus, but not as good as the short story corpus. If I consider both classes, these experiments have a better balance of higher scores for the embedded or interruptive category, and the original narrative category.

4.5 Contributions and discussion

The main contribution of this dissertation is embedded in this chapter: identifying a set of features that can accurately extract narrative levels from raw text. This exercise in feature engineering was heavily influenced by my experience running the narrative level annotation study, and my time annotating texts for narrative levels (for a personal pilot study). Speaking to my annotators about how they made their decision while annotating helped me decide which features to experiment with. Additionally, going through the process of doing annotations myself, helped me think about how I made decisions as to where narrative boundaries occur in text, and think about how narrative levels become either embedded or interrupted.

Specifically, using narrative characteristics classifiers, like story, POV and diegesis, in tandem with character based features, enables accurate extraction of narrative boundaries from long form text. Before this work, it was not possible to detect embedded stories occurring in narrative text. Now you can use my features, and the models I produced, to find where a narrative boundary lies, and the length of the narrative level.

This work can be improved. Modifications to my extractor to detect multiple levels of embedded and interruptive narrative is a clear next augmentation. Additionally, research must be done on extracting narrative levels from the scripts, since my extractors failed to properly learn and classify these narratives. Finally, research into clustering each span of embedded or interruptive text into distinct narrative levels should be conducted. Now the detector only knows there is a new level; it cannot decide which spans belong to which levels, and I think a clustering technique that uses topic and character modeling will be a fruitful path for seeding new work.

Chapter 5

Automatic extraction of stories from text[8]

Story detection is the task of determining if a unit of text contains a story. Prior approaches achieved a maximum performance of 0.66 F_1, and did not generalize well across different corpora. Here I present a new state-of-the-art detector that achieves a maximum performance of 0.75 F_1 (a 14% improvement), with significantly greater generalizability than previous work. In particular, the detector achieves performance above 0.70 F_1 across a variety of combinations of lexically different corpora for training and testing, as well as dramatic improvements (up to 4,000%) in performance when trained on a small, disfluent data set. The new detector uses two basic types of features–ones related to events, and ones related to characters–totaling 283 specific features overall; previous detectors used tens of thousands of features, and so this detector represents a significant simplification along with increased performance. At the end of this chapter I will talk about prior work I did on implementing previous story classifiers, and evaluating them on common data sets to determining which approach was most effective.

[8] This chapter was adapted from two articles I published: First an article in the Proceedings of the Conference on Empirical Methods in Natural Language Processing (Eisenberg, 2017), and second an article at the Workshop on Computational Models of Narrative (Eisenberg, 2016).

5.1 Motivation

Understanding stories is a long-held goal of both artificial intelligence and natural language processing. Stories can be used for many interesting natural language processing tasks, and much can be learned from them, including concrete facts about specific events, people, and things; commonsense knowledge about the world; and cultural knowledge about the societies in which I live. Applying NLP directly to the large and growing number of stories available electronically, however, has been limited by our inability to efficiently separate story from non-story text. For the most part, studies of stories per se has relied on manual curation of story data sets (Mostafazadeh, 2016), which is, naturally, time-consuming, expensive, and doesn't scale. These human-driven methods pay no attention to the large number of stories generated daily in news, entertainment, and social media.

The goal of this work is to build and evaluate a high performing story detector that is both simple in design and generalizable across lexically different story corpora. Our definition of story can be found in §5.1.2, and is based on definitions used in prior work on story detection. Previous approaches to story detection have relied on tens of thousands of features (Ceran, 2012; Gordon, 2009), and have used complicated pre-processing pipelines (Ceran, 2012). Moreover these prior systems, while clearly important advances, did not, arguably, include features that captured the "essence" of stories. Furthermore, these prior efforts had poor generalizability, i.e. when trained on one corpus, the detectors perform poorly when tested on a different corpus. Building on this prior work, I begin to address these shortcomings, presenting a new detector that has many orders of magnitude fewer features than used previously, significantly improved cross corpus performance, and higher F_1 on all training and testing combinations.

5.1.1 Task

Our goal is to design a system that can automatically decide whether or not a paragraph of text contains a story. A paragraph contains a story if any portion of it expresses a significant part of a story, including the characters and events involved in major plot points. Corpora used in prior work included Islamic Extremist texts (Ceran, 2012), and personal web blog posts (Gordon, 2009), which were both annotated at this level of granularity. In this chapter I test combinations of new features on both of these corpora. Once I determined the best-performing feature set, I ran experiments using those features to evaluate its generalizability across corpora.

5.1.2 What is a story?

Author E.M. Forster said "A story is a narrative of events arranged in their time sequence" (Forster, 2010). A more precise definition, of our own coinage, is that a narrative is a discourse presenting a coherent sequence of events which are causally related and purposely related, concern specific characters and times, and overall displays a level of organization beyond the commonsense coherence of the events themselves. In sum, a story is a series of events effected by animate actors. This reflects a general consensus among narratologists that there are at least two key elements to stories, namely, the plot (fabula) and the characters (dramatis personae) who move the plot forward (Abbott, 2008). While a story is more than just a plot carried out by characters–indeed, critical to 'storyness' is the connective tissue between these elements that can transport an audience to a different time and place–here I focus on these two core elements to effect better story detection.

5.1.3 Chapter outline

I begin by discussing prior work on story detection (§5.2). Then I introduce the new detector (§5.3), which relies on simple verb (§5.3.1) and character (§5.3.2) features. I tested the detector on two corpora (§5.3.3)—one of blog posts and one of Islamist Extremist texts—using an SVM model to classify each paragraph as to whether or not it contains a story (§5.3.4). I conducted an array of experiments evaluating different combinations and variants of our features (§5.4). I detail the use of undersampling for the majority class (§5.4.1), as well as the cross validation procedure (§5.4.2). I present both the results of the single corpus experiments (§5.4.3) and the cross-corpus and generalizability experiments (§5.4.4). Then I will do a review of work I did before I built my verb and character based story classifier. This work is on reimplementing the first two story classifiers, and how I evaluated their performance (§5.5). Then I conclude with a list of contributions and discussion of future directions (§5.6).

5.2 Related work

There have been three major contributions to the study of automatic story detection. In 2009, Gordon and Swanson developed a bag-of-words-based detector using blog data (Gordon, 2009). They annotated a subset of paragraph-sized posts in the Spinn3r Blog corpus for the presence of stories, and used this data to train a confidence weighted linear classifier using all unigrams, bigrams, and trigrams from the data. Their best F_1 was 0.55. This was an important first step in story detection, and the annotated corpus of blog stories is an invaluable resource.

In 2012, Corman et al. developed a semantic triplet-based detector using Islamist Extremist texts (Ceran, 2012). They annotated paragraphs of the CSC Islamic Extremist corpus for the presence of stories, and used this data to train an SVM with a variety of features including the top 20,000 tf-idf tokens, use of stative verbs, and agent-verb-patient triplets ("semantic triplets," discussed in more detail below in §3.1). Their best performing detector in that study achieved 0.63 F_1. The intent of the semantic triplet features was to encode the plot and the characters. These features were intended to capture the action of stories, but the specifics of the implementation was problematic: each unique agent-verb patient triplet has its own element in the feature vector, and so this detector was sensitive primarily to the words that appeared in stories, not generalized actions or events.

Although Corman's detector has a higher F_1 than Gordon's, it was not clear which one was actually better; they were tested on different corpora. I compared the two detectors by reimplementing both, confirmed the correctness of the reimplementations, and running experiments where each detector was trained and tested on the corpora (Eisenberg et al., 2016). After these experiments, I showed that Corman's detector had better performance on the majority of experiments. Some of the results of these experiments are shown in Table 5.5. I also slightly improved the performance of Corman's detector to 0.66 F_1. In addition I reported results investigating the generalizability of the detectors; these results showed that neither the Gordon nor the Corman detectors generalized across corpora. I ascribed this problem to the fact that the features of each detector were closely tied to the literal words used, and did not attempt to generalize beyond those specific lexical items.

In terms of domain independence, I surveyed other discourse related tasks to see how generalization across domains has been achieved. For example, Braud (2014) achieved domain independence in the identification of implicit relations between discourse units by training their system on both natural and synthetic data, weighting the influence of the two types (Braud, 2014). Jansen (2014), as another example, demonstrated domain independence on the task of non-factoid question answering by using both shallow and deep discourse structure, along with lexical features, to train their classifiers (Jansen 2014). Thus, domain independence is certainly possible for discourse related tasks, but there does not yet seem to be a one-size-fits-all solution.

5.3 Developing the detector

In contrast to focusing on specific lexical items, our implementation focuses on features which I believe capture more precisely the essence of stories, namely, features focusing on (a) events involving characters, and (b) the characters themselves. Figure 5.1 contains a block diagram which represents the pipeline for the process of extracting features from paragraphs of raw text.

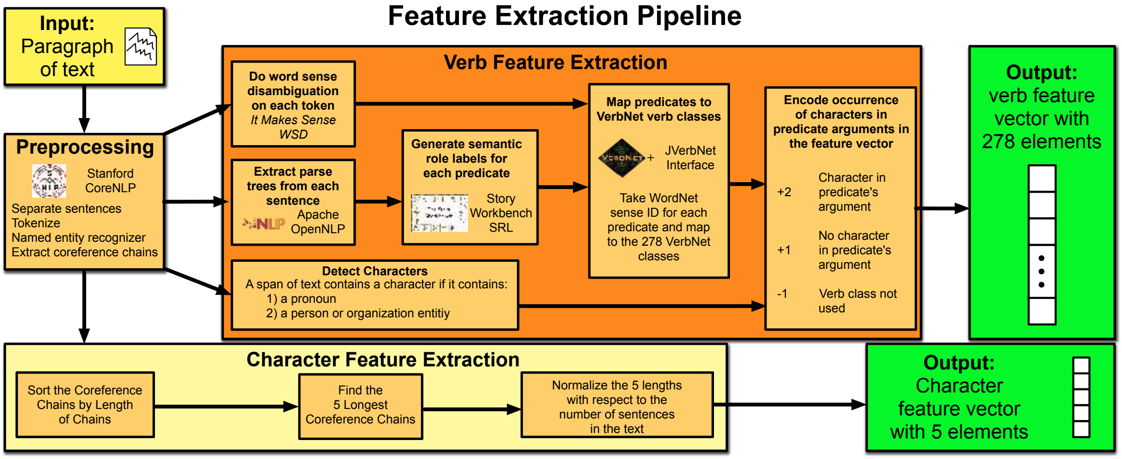

Feature Extraction Pipeline
Input: Paragraph of text
Preprocessing
Stanford CoreNLP
Separate sentences
Tokenize
Named entity recognizer
Extract coreference chains
Verb Feature Extraction
Do word sense disambiguation on each token
It Makes Sense WSD
Extract parse trees from each sentence
Apache OpenNLP
Generate semantic role labels for each predicate
Story Workbench SRL
Map predicates to VerbNet verb classes
JVerbNet Interface
Take WordNet sense ID for each predicate and map to the 278 VerbNet classes
Encode occurrence of characters in predicate arguments in the feature vector
+2 Character in predicate's argument
+1 No character in predicate's argument
-1 Verb class not used
Detect Characters
A span of text contains a character if it contains:
1) a pronoun
2) a person or organization entitiy
Output: verb feature vector with 278 elements
Character Feature Extraction
Sort the Coreference Chains by Length of Chains
Find the 5 Longest Coreference Chains
Normalize the 5 lengths with respect to the number of sentences in the text
Output: Character feature vector with 5 elements

Figure 5.1 Diagram of the story extractor feature extraction pipeline.

5.3.1 Verb features

Verbs are often used to express events. I use this fact to approximate event detection in a computationally efficient but still relatively accurate manner. The first part of each feature vector for a paragraph comprises 278 dimensions, where each element of this portion of the vector represents one of the 278 verb classes in VerbNet (Schuler, 2005). The value of each element depends on whether a verb from the associated verb class is used in the paragraph. Each element of the vector can have three values: the first value represents when a verb from the element's corresponding verb class is used in the paragraph and also involves a character as an argument of the verb. The second value represents when a verb from the verb class is used, but there are no characters involved. The third value represents the situation where no verbs from the verb class are used in the paragraph.

For clarity, here are the steps of the verb feature extraction pipeline:

1. Split each paragraph into tokens, assign part of speech tags, and split the text into sentences, all using Stanford CoreNLP (Manning et al., 2014).
2. Parse each sentence with OpenNLP (Apache Foundation, 2017).
3. Label each predicate with its semantic roles using the SRL from the Story Workbench (Finlayson, 2008, 2011).
4. Disambiguate the Wordnet sense (Fellbaum, 1998) for each open-class word using the It Makes Sense WSD system (Zhong, 2010), using the Java WordNet Interface (JWI) to load and interact with WordNet (Finlayson, 2014).
5. Assign one of 278 VerbNet verb classes to each predicate, based on the assigned Wordnet sense, and using the jVerbnet library to interact with VerbNet. (Finlayson, 2012).
6. Determine whether the arguments of each predicate contains characters by using the Stanford Named Entity Recognizer (Finkel et al., 2005) and a gendered pronoun list.

I considered an argument to involve a character if it contained either (1) a gendered pronoun or (2) a named entity of type Person or Organization. I treated organizations as characters because they often fulfill that role in stories: for example, in the Extremist stories, organizations or groups like the Islamic Emirate, Hamas, or the Jews are agents or patients of important plot events. The verb features were encoded as a vector with length

278, each entry representing a different VerbNet verb class with three possible values: the verb class does not appear in the paragraph; the verb class appears but does not involve characters; or the verb class appears and a character is either an agent, patient, or both.

The verb features represent the types of events that occur in a paragraph, and whether or not characters are involved in those events. This is a generalized version of the semantic triplets that Corman et al. used for their story detector (Ceran, 2012), where they paired verbs with the specific tokens in the agent and patient arguments. The disadvantage of Corman's approach was that it led to phrases with similar meaning being mapped to different features: for example, the sentences "Mike played a solo" and "Trey improvised a melody" are mapped to different features by the semantic triplet based detector, even though the meaning of the sentences are almost the same: a character is performing music. On the other hand, in our approach, when I extract verb feature vectors from these sentences, both result in the same feature value, because the verbs played and improvised belong to the performance VerbNet class, and both verbs have a character in one of their arguments. This allow a generalized encoding of the types of action that occurs in a text.

5.3.2 Character features

Our second focus is on character coreference chains. A coreference chain is a data structure that represents mentions of an entity in a text. Consider the following text: "Josh went to the beach. He got a sunburn." Here the pronoun "he" is used to refer to the character "Josh". A coreference chain keeps track of all the links between pronouns, and other references, with the characters they refer to.

Characters, as discussed previously, are a key element of stories. A character must be present to drive the action of the story forward. I hypothesize that stories will contain longer coreference chains than non-stories. To encode this as a feature, I calculated the normalized length of the five longest coreference chains, and used those numbers as the character features. I computed these values as follows:

1. Extract coreference chains from each paragraph using Stanford CoreNLP coreference facility (Clark, 2016).
2. Filter out coreference chains that do not contain a character reference as defined in the Verb section above (a named entity of type Person or Organization, or a gendered pronoun).

3. Sort the chains within each paragraph with respect to the number of references in the chain.
4. Normalize the chain lengths by dividing the number of referring expression in each chain by the number of sentences in the paragraph.

The normalized chain lengths were used to construct a five-element feature vector for use by the SVM. I experimented with different numbers of longest chains, anywhere from the single longest to the ten longest chains. Testing on a development set of 200 Extremist paragraphs revealed using the five longest chains produced the best result.

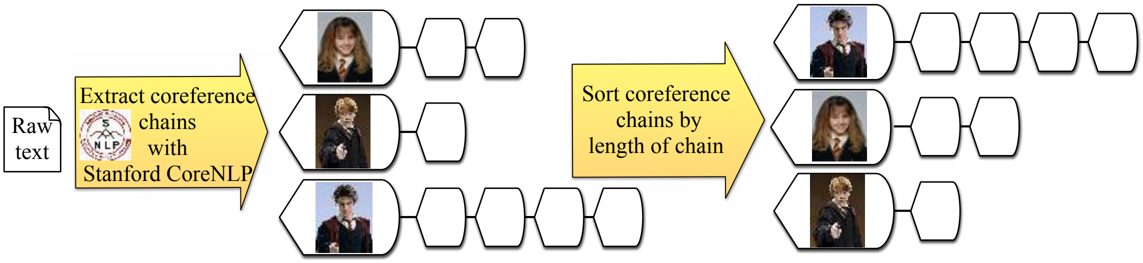

Raw text
Extract coreference chains with Stanford CoreNLP
Sort coreference chains by length of chain

Figure 5.2 Visualization of character feature extraction pipeline

5.3.3 Corpora

As noted, I used two corpora that were annotated by other researchers for the presence of stories at the paragraph level. The CSC Islamic Extremist Corpus comprises 24,009 paragraphs (Ceran, 2012), of which 3,300 were labeled as containing a story. These texts recount Afghani and Jihadi activities in the mid-2000's in a variety of location around the world. This corpus was originally used to train and test Corman's semantic-tripletbased story detector. The web blog texts come from the ICWSM 2009 Spinn3r Dataset (Burton et al., 2009). The full data set contains 44 million texts in many languages. Gordon and Swanson (2009) annotated a sample of 4,143 English texts from the full data set, 201 of which were identified as containing stories. This corpus was originally used to train and test Gordon's bag-of-words based detector. Most of the texts in the blog corpus are no more than 250 characters, roughly a paragraph. The distribution of texts can be seen in Table 5.1.

Corpus	Story	Non-story
Extremist	3,300	20,709
Blog	201	3,942

Table 5.1 Distribution of stories in extremist and blog corpora

5.3.4 SVM machine learning

I used the Java implementation of LibSVM (Chang and Lin, 2011) to train an SVM classifier with our features. The hyper-parameters for the linear kernel were $\gamma = 0.5$, $\nu = 0.5$, and $c = 20$.

5.4 Experiments and results

The results of the new experiments are shown in Table 5.3. I report precision, recall, and F_1 relative to the story and non-story classes. I performed experiments on three feature sets: the verb features alone (indicated by Verb in the table), character features alone (indicated by **Char**), and all features together (**Verb+Char**). I conducted experiments ranging over three corpora: the Extremist corpus (**Ext**), the blog corpus (**Web**), and the union of the two (**Comb**). These results may be compared with the previously best performing detector, namely, Corman's semantic triplet based detector (Ceran, 2012), as tested by us in prior work (Eisenberg, 2016), and shown in Table 5.2.

Training	Testing	Prec.	Recall	F1
Ext	**Ext**	0.77	0.57	**0.66**
Ext	**Web**	0.23	0.37	0.28
Ext	**Comb**	0.43	0.41	0.32
Web	**Web**	0.66	0.31	0.43
Web	**Ext**	0.59	0.003	0.01
Web	**Comb**	0.59	0.01	0.01
Comb	**Ext**	0.62	0.51	0.43
Comb	**Web**	0.36	0.49	0.3
Comb	**Comb**	0.64	0.47	0.46

Table 5.2 Results for the Corman semantic triple based detector from Eisenberg (2016).

			Not Story			Story		
Features	**Training**	**Testing**	Prec.	Recall	F1	Prec.	Recall	F1
Verb	**Ext**	**Ext**	0.73	0.81	0.77	0.78	0.7	0.74
Verb	**Web**	**Web**	0.69	0.75	0.72	0.73	0.66	0.69
Char	**Ext**	**Ext**	0.3	0.27	0.21	0.52	0.74	0.55
Char	**Web**	**Web**	0.67	0.68	0.67	0.67	0.65	0.65
Verb+Char	**Ext**	**Ext**	0.73	0.81	0.77	0.79	0.7	**0.74**
Verb+Char	**Ext**	**Web**	0.68	0.8	0.73	0.75	0.63	0.69
Verb+Char	**Ext**	**Comb**	0.7	0.77	0.73	0.75	0.67	0.71
Verb+Char	**Web**	**Web**	0.71	0.76	0.72	0.74	0.68	0.7
Verb+Char	**Web**	**Ext**	0.5	0.82	0.62	0.5	0.18	0.27
Verb+Char	**Web**	**Comb**	0.53	0.79	0.64	0.6	0.4	0.41
Verb+Char	**Comb**	**Ext**	0.74	0.81	0.77	0.79	0.71	**0.75**
Verb+Char	**Comb**	**Web**	0.68	0.74	0.7	0.72	0.64	0.67
Verb+Char	**Comb**	**Comb**	0.72	0.81	0.76	0.79	0.68	0.73

Table 5.3 Results of new detectors across all corpora from Eisenberg (2017).

5.4.1 Undersampling

In each of the new experiments, I undersampled the non-story class before training (Japkowicz, 2000). Undersampling is a technique used to help supervised machine learning classifiers learn more about a class that has a significantly smaller number of examples relative to an alternative. In our case, non-story labels outnumbered story labels by a factor of 7 overall. Extremist story paragraphs are only 15.9% of the total annotated paragraphs in that set, and in the blog corpus stories were only 4.9% of the paragraphs. To prevent the detector from being over trained on non-story paragraphs, I thus reduced the size of the nonstory training data to that of the story data, by randomly selecting a number of non-story texts equal to the number of story texts for training and testing.

5.4.2 Cross-validation

I used three versions of cross validation for the new experiments, one for each experimental condition: training and testing on a single corpus; training on a single corpus and testing on the combined corpus; or training on the combined corpus and testing on a single corpus. These procedures are the same as in our previous work (Eisenberg, 2016). I performed undersampling before cross validation, so when I am explaining how to divide up the story and non-story texts into cross validation folds, this refers to the full set of story texts and the set of non-story texts that was randomly selected to equal the number of story texts. For all experiments with cross validation, I use ten folds.

Train and Test on a Single Corpus: If the training and testing corpus is the same, divide up the stories into ten subsets of equal size, and the undersampled non-stories into ten subsets of equal size. For each fold of cross-validation a different story set and non-story set (of the same index) are used as the testing set and the remaining nine are used for training.

Train on Combined, Test on Single: If the training is done on the combined corpus, and the test corpus is either the web blog or Extremist corpus, which I will refer to as the single test corpus, first divide the stories from the single test corpus into ten equal sized sets, and then divide up that corpus's non-stories into ten equal sets. For each fold of cross validation a different story set and non-story set (of the same index) from the single test corpus are used as the testing set and the remaining nine are used for training. The texts from the other corpus (the corpus that is not the single test corpus), are undersampled and added to all ten folds of training.

Train on Single, Test on Combined: If training is done on a single corpus, and the test corpus is the combined corpus, first divide the stories from the single training corpus into ten equal sized sets, and the undersampled non-stories from the single training corpus into ten equal sized sets. For each fold of cross validation a different story set and non-story set (of the same index) from the single training corpus are used as the testing set and the remaining nine are used for training. The texts from the other corpus (the corpus that is not the single training corpus), are undersampled and added to all ten folds of testing.

5.4.3 Single corpus experiments

For every experiment that used only a single corpus, the best feature set included both the verb and character features, achieving up to 0.74 F_1 when trained and tested on the Extremist corpus. This is the new state-of-the-art, about 12.6% greater than the performance of Corman's detector when trained and tested on the same corpus (0.66 F_1).

When the detector uses only verb features it achieves an F_1 of 0.74 on the Extremist corpus, only 0.002 lower than the detector using all the features. Interestingly, the detector achieves 0.55 F_1 using only the five character features, which is respectful given such a small feature set. To put this in perspective, the Corman detector (Ceran, 2012) uses more than 20,000 features, and achieves an F_1 of 0.66. Thus I was able to achieve 83% of the performance of the Corman detector with 4,000 times fewer features.

When training and testing on the blog corpus, the detector using all the features achieved 0.70 F_1, a 74% increase from the Corman detector's 0.425 F_1. This is the best performing model on the blog corpus, from any experiment to date. The detector using only verb features achieves 0.74 F_1, which is only slightly worse than when both sets of features are used. When I trained using only the character features, the system achieves 0.65 F_1, which is still 54% higher than when the Corman detector is trained and tested on the blog corpus.

In the single corpus experiments, the detectors that I trained and tested on the Extremist paragraphs have higher performance than those trained on the web blogs, except for when I use only the five character features. A possible reason for this is the Stanford NER may not be recognizing the correct named entities in the Extremist texts, which contain many non-Western names, e.g., *Mujahidin*, *Okba ibn Nafi*, or *Wahid*. However, when I include the verb features, the detectors trained on the Extremist texts achieve better performance. I believe this is partially due to the greater number of stories in the Extremist corpus, and their increased grammatical fluency. The Extremist corpus is actually well written compared to the blog corpus, the latter of which contains numerous fragmentary and disjointed posts.

5.4.4 Cross corpus experiments

I show the generalizability of our best performing detector (that including both verb and character features) by training it on one corpus and testing it on another. When I trained the detector on the Extremist texts and tested on the blog texts, it scores a 0.68 F_1. This is 142% improvement over Corman's detector in the same setup (0.28 F_1), and is a higher F_1 than the previous state-of-the-art on any single corpus test. When I trained the detector on the Extremist corpus and tested on the combined corpus, it achieved 0.71 F_1, which is an 121% increase from Corman's detector in the equivalent setup.

For the detector trained on the blog corpus and tested on the Extremist corpus, the detector that uses both verbs and character features achieves an 0.27 F_1, which is a 2,600% increase over the Corman detector's 0.01 F_1 in this same setup. While 0.27 F_1 can by no means be called good performance, it is significantly better than the Corman detector's performance on this task, and so demonstrates significantly better generalizability. As seen in our experiments, detectors trained on only the blog corpus do not perform as well as detectors trained on the Extremist corpus. I suspect that this is partially due to the disfluent nature of the blog corpus, which includes many fragmentary sentences, grammatical errors, and slang, all of which are difficult for the NLP pipeline to handle.

Note that I performed no cross validation in the above experiments where I trained the detector on the Extremist corpus and tested on the blog corpus, or vice versa, because in these cases the training and testing sets have no intersection.

The cross corpus experiment with the largest percent increase is for the verb and character detector trained on the blog corpus and tested on the combined corpus. The new detector's F_1 is 0.41, a 4,000% increase from the Corman detector's 0.01 F_1 on this task. Although a 0.41 F_1 is also not good, this is a massive improvement over previous performance. This is further evidence that our verb and character feature based detector is significantly more generalizable than Corman's approach.

The remaining five cross corpus experiments involved the combined corpus. In this case, our detector out-performed Corman's detector. Of special note is the detector trained on the combined corpus and tested on the Extremist corpus. It achieved 0.75 F_1, which is 0.01 points of F_1 higher than our best single corpus detector, which was trained and tested on the Extremist corpus. This isn't a substantial increase in performance, but it suggests that information gleaned from the blog corpus does potentially—albeit marginally—help classification of the Extremist texts.

5.5 Prior work on story classification[9]

Before my work there were only two approaches to story classification: one developed by Gordon et al. at USC (Gordon, 2009), the other by Corman et al. at ASU (Corman 2012). Both the Gordon and Corman classifiers leverage supervised machine learning algorithms trained on large annotated datasets, and both use linguistic features to separate story from non-story text. Gordon's classifier uses unigram frequencies to classify stories. This classifier was originally tested on the ICWSM 2009 Spinn3r blog dataset, which contains personal stories that were posted to blogs in 2009 (Burton 2009). Corman's classifier, on the other hand, focuses on verbs and their patient and agent arguments (semantic triplets). It also considers the unigram frequencies and density of various features such as part of speech tags, named entities, and stative verbs. Corman's classifier was originally tested on the CSC corpus of Islamic Extremist texts, in which each paragraph was annotated as either *story*, *exposition*, *supplication*, *religious verse*, or *other*.

In the following subsection (§7.1.1) I will take a detailed look at the Gordon and Corman story classifiers. In a previous publication (Eisenberg 2016), I reimplemented both classifiers, and evaluated their performance on their original test corpus to confirm the results of the original studies. The following sections will take an in-depth look at the design on the previous classifiers, by show how I reimplemented them.

Next, I tested the reimplemented classifiers across the other corpora to gain insight into which classifier had better performance, and in turn decide which classifier had a better

[9] This section was adapted from an article I published in the Proceedings of the Seventh International Workshop on Computational Models of Narrative (Eisenberg, July 2016).

approach to the problem of story classification. In §7.1.2 I will discuss the results of these experiments, and how it led to the creation of the story extractor built for the work in Chapter 5.

5.5.1 Design of previous story classifiers

5.5.1.1 Reimplemented Gordon story classifier

The philosophy of Gordon's classifier is stories are made up of words, and so to find stories one should look for story-relevant words. Gordon's story classifier uses unigram features (Gordon 2009) and so makes associations based on what words appear in stories vs. non-stories (a "bag of words" approach). The features (words) extracted from each text in the training set are used to train a confidence weighted linear classifier (Dredze 2008). This is like perceptron learning (Ng, 1997; Rosenblatt 1958), but it has augmentations which can improve how it learns NLP features. Although the confidence weighted linear classifier can be better suited than the perceptron for classifying certain NLP phenomena, I did not find that to be true in our story classification experiments. To train the confidence weighted linear classifier each word is run through the classifier one time (one epoch); it is uncertain how many epochs of training Gordon used, but I found that performance did not improve significantly by training this classifier for more epochs. After training, the confidence weighted linear classifier has assigned each individual word a weight that represents how relevant it is in classifying text as a story. Weights closer to zero imply that the word occurs in both stories and non-stories with similar frequency, while weights far from zero imply that the relevant word appears correlates with one of the two classes (stories or non-stories).

To classify a document, its words are extracted in the same manner as they are in the training set. Then the feature counts are capped: all counts above 7 are brought down to 7. Finally, the feature values are normalized to values between 0 and 1.

There are two parts to our reimplementation: the feature extractor and the confidence weighted linear classifier. The feature extraction pipeline is built in Java, but the classifier is written in the Go programming language, in order to make use of the *gonline library*[10], which is a library of online machine learning algorithms written in Go. I used the goline library because I could not find a usable version of the confidence weighted linear classifier in Java. I modified the gonline library to produce more fine-grained error statistics, and to suppress false parser errors. For the feature extractor, I use some of the same text preprocessing that Gordon provided[11]. I use the same regular expressions from his Python code to break up clitics, punctuation, and irregular characters. Then I feed the filtered data through the Stanford Tokenizer (Manning, 2014) to turn each document into a stream of tokens. This stream of tokens is used for the unigram counting.

5.5.1.2 Improved Gordon story classifier

During reimplementation I experimented with replacing the confidence-weighted linear classifier with traditional single layered biased perceptron network [13, 16]. This "Perceptron" version of the Gordon classifier scored an F_1 measure 5 points higher than

[10] https://github.com/tma15/gonline

[11] https://github.com/asgordon/StoryNonstory

our implementation of the confidence weighted Gordon classifier. This is interesting because the confidence weighted linear classifier has been reported as better at classifying data from the NLP domain [5]. In this case, not only was the Perceptron classifier easier to develop than the confidence weighted one, it also had better performance. These findings are expanded upon in §3.2. In our final implementation of this improved classifier, I trained the Perceptron for 10 epochs and used a learning rate of 0.005. I settled on these hyperparameters via tuning and experimentation. Additionally, for each epoch, I randomized the order that the training vectors are shown to the Perceptron, because Perceptrons are known to be quite sensitive to the order that the training examples are learned.

The final difference between the original implementation and our improved implementation is that our encoding scheme for the frequencies is slightly different. Our augmentation is a smoothing of all the feature values: 0.07 (roughly 1/14) is added to each feature value. I did this to guarantee that the weights for each feature will be updated during the Perceptron learning, as features with value 0 do not contribute to the learning (in our implementation).

5.5.1.3 Reimplemented Corman story classifier

Corman's semantic triplet based classifier is trained on a wide variety of features "of varying semantic complexity" so it requires a robust linguistic pipeline to facilitate feature extraction (Ceran, 2012). Some of the features are based on lexical properties: the densities of the 30 Penn Tree Bank part of speech tags (Santorini, 1990), stative verbs, and person, location, and organization named entities. The term frequency inverse document frequency (tf-idf) (Salton, 1975) measure is calculated for each word in each document of the training corpus and, in this implementation, the words with the highest 20,000 tf-idf measures are features. The final feature is the semantic triplet. These are triplets of each verb with their respective patient and agent arguments. Sometimes this feature is actually a four-tuple where the fourth term is a locational preposition. The lexical features and triplets are extracted from each document in the training set and used to train a support vector machine (SVM) with an RBF function (Keerthi 2003).

Corman's 2012 paper gives a high-level description of how these features are extracted from each document. Our classifier differs in a few non-trivial ways:

- I do not use the Illinois Semantic Role Labeler (Punyakanok, 2008). I use a semantic role labeler built from scratch in our lab, which is included in the Story Workbench linguistic annotation tool (Finlayson 2008 & 2011). I used our own tool because the Illinois SRL is quite heavyweight: it requires installation of the Illinois Curator Serve (Clarke, 2012) and MongoDB[12].
- I do not do coreference resolution to replace the pronouns with their corresponding referent entity.
- I do no alias standardization. It was unclear how this should be carried out, since the named entity tagging is run on each token, and there was no explanation for how multi-word named entity boundaries were determined.

[12] https://www.mongodb.com/

- I performed no spell checking on our named entities for the same reason as the alias standardization.
- I only use the Stanford named entity recognizer (Finkel, 2005) and the Illinois named entity tagger (Ratinov, 2009). I do not use the Open Calais named entity recognizer[13] because I wanted the classifier to run without needing to query a limited resource on the internet.

I removed three portions of the text preprocessing, used a different SRL, and one fewer NER than Corman's original implementation. Even though I built a less complicated version of Corman's classifier, it performed approximately the same as (or even slightly better than) the original.

To extract semantic triplets, I first extract the parse tree for a sentence. I pass this parse tree to the SRL, which extracts all the predicates and their arguments. I take the lemmatized predicates and search for a VerbNet (Schuler, 2005) category based on the number of arguments in Propbank (Kingsbury 2013); I take an exact match if there is one, but take the closest match otherwise. Failing this, I return the lemma of the word from Wordnet. I follow the following rules, set forth by Ceran et al. in their paper:

- If our object has multiple verbs, it is complex. I create new triplets for the object recursively and assign a pointer to the new triplets as the object for this triplet.
- If the SRL doesn't return Arg1, I substitute Arg2 for the object.

[13] https://en.wikipedia.org/wiki/Calais_(Reuters_product)

- If Arg2 is a location preposition, I include it as a fourth element.
- If Arg2 is a preposition, I use Arg1 as the subject and Arg2 as the object.

Due to some shortcomings in our SRL I may find that some or all of the arguments for a given predicate are null; in the event that some of our predicates are null, I simply tag the remaining slot (either subject or object) with a "-1" indicating a lack of an argument. If all of our arguments are null I attempt to find the closest noun behind the predicate, which I assign as the subject, and the closet noun ahead of the predicate, which I assign as the object. As above if I result in a null argument from finding the closet nouns I tag those slots with "-1".

To use the triplets as a feature, I extract all the possible subjects, objects, and location prepositions for a given verb or verb category from across the entire corpus. Then I assign each verb and verb category a specific index. These indices are determined by a simple alphabetical sort: I check every document's triplets and assign it a "1" at a given index if it has that verb or that argument for the verb and a "0" otherwise. These features are used to train a SVM that uses a radial bias function (RBF) kernel function and a soft margin C of 10,000, which is a relatively standard setting (I am not sure what soft margin parameter was used by Corman). This produces a model that can classify whether a text contains a story. To test text on the model, the same types of features extracted in training, are extracted from the test document. The feature values are used with the model to obtain a classification value.

5.5.2 Results

5.5.2.1 Experimental results of the reimplemented story classifiers

To show that the reimplemented classifiers behave the same way as the originals, they were trained and tested on the same data sets as in the original studies. The Gordon confidence weighted (CW) linear classifier was trained and tested on the Spinn3r Web blog Corpus. I used the same texts that Gordon used in his study. As can be seen in Table 5.4, in terms of F_1 our Gordon CW reimplementation performs almost 8 points worse than what Gordon reported. Our Gordon CW classifier has a precision of 0.475, recall of 0.5, and an F_1 of 0.471. This could be because Gordon used a different version of the CW algorithm than I did, but it is unclear why our implementation performs so differently. Although our modified Gordon Perceptron is arguably simpler than our Gordon CW, it has a higher F_1 by almost 5 points. The Gordon Perceptron's F_1 is 0.522, which is almost 3 points less than the performance Gordon reported. I cannot say that the Gordon CW is a good reimplementation of the original Gordon classifier since the F_1 measures are significantly different. On the other hand, the performance of our Perceptron and Gordon's original classifier are quite similar, which is encouraging.

Corman et al. trained and tested their classifier on the CSC Islamic Extremist Corpus. I used the same texts during this experiment on the reimplementation. As can be seen in Table 5.5, our reimplementation of Corman's classifier performs similarly to the original version. The reimplementation scored a precision of 0.773, recall of 0.573, and F_1 of 0.658, which compares favorably with the originally reported results.

In terms of F_1 our performance scored slightly higher than that of the original system. This may be due to the differences in the preprocessing pipeline and the triplet extraction, as discussed in §2.3. Nevertheless, I take the similarity of the results as evidence that our reimplementation is roughly faithful to the original.

System	Training	Testing	Prec.	Recall	F1
Gordon Reported	Web	Web	0.66	0.48	0.55
Gordon CW	Web	Web	0.48	0.5	0.47
Gordon Perceptron	Web	Web	0.65	0.46	0.522

Table 5.4 Results for Gordon classifiers on the web blog corpus.

System	Training	Testing	Prec.	Recall	F1
Corman reported	Ext	Ext	0.73	0.56	0.63
Corman	Ext	Ext	0.77	0.57	0.66

Table 5.5 Results for Corman classifiers on the extremist corpus.

5.5.2.2 Cross-testing the classifiers

Because I have both classifiers and both datasets, I performed experiments to compare how each classifier performs on the other data, as well as on both datasets simultaneously. I trained and then tested both our reimplemented and improved classifiers on all combinations of the three corpora. For the cross-tests of the Gordon Classifier I use the Gordon Perceptron. As shown in the previous section, the Gordon Perceptron performs closer to the original Gordon Classifier than our reimplementation of the Gordon CW. Using the Gordon Perceptron allows for a more accurate comparison of the classifier than our Gordon CW implementation. For each of the cross-tests, the corpora go through 10-fold cross validation to generate the training and testing sets, as follows:

- If the training and testing corpora are the same, divide up the stories into ten subsets of equal size, and the not stories into ten sets of equal size. For each fold of cross validation a different story set and the not story set (of the same index) are used as the testing set and the remaining 9 are used for training.
- If the training is done on the combined corpus, and the test corpus is either the web blog or extremist corpus, which I will refer to as the single corpus, first divide the stories into ten equal sized sets, and then divide up that corpus' not stories into ten equal sets. First divide the stories in the single corpus into ten equal sets, and the not stories of the same corpus into ten equal sets. These can be split into the training and testing sets the same as in the previous situation. Additionally, the whole other corpus, the one that is not the single corpus, is added to the training set.
- If training is done on a single corpus, and the test corpus is the combined corpus, first break up the stories and not stories of the single corpus each into ten equal subsets. Assign them to the training and testing set as in the first situation. Then add the whole other corpus, the one that is not the single corpus, to the testing set.

The results can be seen in Table 5.2. Macro-averages for precision, recall and F_1 are reported for each experiment. I chose to report macro averaging since it was less sensitive to outliers and atypical models. Unless stated otherwise, the F_1 measures reported in this study are relative to the story class. Using story as the label to classify correctly is a good way to measure performance for this task, since the overall goal is to produce a system that can accurately identify stories.

5.5.3 Discussion

The Corman classifier has its best performance when tested and trained on the extremist corpus, while the Gordon Perceptron does best on the web blog corpus. Both classifiers perform best when tested and trained on the corpus that it was tested on in their original studies. The Corman classifier scores best across all the experiments: 0.66 F_1 for when it is trained and tested on the extremist corpus. Yet, the Corman classifier F_1 is 10 points worse than the Gordon Classifier when tested and trained on the web blog corpus. This suggests that the Corman classifier has a harder time learning from the web blog corpus than the extremist corpus.

The Gordon classifier performed best when trained and tested on the web blog corpus. The F_1 measure for this experiment was 0.522, which is about 3 percentage points lower than the best result Gordon reported, and I hypothesize that this is because differences in our CW classifier implementations

When the Corman classifier, with the Extremist model, is tested on the Web blog corpus it only has an F_1 of 0.283. This poor performance is because the model has not been trained to recognize the type of stories in the Web blog corpus. The stories in the Extremist corpus are typically 3rd person, second hand accounts, not 1st person personal accounts. Even though the Web blog corpus has significant syntactic irregularity, it contains a different type of story than that present in the Extremist corpus, so it is still useful for model training.

None of the cross-tests perform as well as the tests on the original corpus. The only cross test that comes close to the original performance is when the Corman classifier is trained on the combined corpus and tested on either the extremist or combined corpora, respectively their F_1 measures are 0.647 and 0.632. This makes sense, because the training set mostly contains examples from the extremist corpus, thus the model is more heavily influenced by that corpus. The extremist corpus has about five times more texts than the web blog corpus, hence the training set will have five times more texts from the extremist corpus. In effect, the combined model is quite similar to the extremist model.

The two weakest of all the cross tests are when the models are trained using the Web blog data and tested on the combined corpus. When trained on the Web blog corpus and tested on the combined corpus the F_1 measure Gordon's F_1 is 0.014, and Corman's F_1 is 0.007. Another particularly weak experiment was when the Corman classifier is tested on the web blog corpus but tested on the extremist corpus. The F_1 for this experiment is 0.007. This suggests that the Web blog data does not generalize well to the Extremist set. Although, the Corman classifier has experiments with the highest F_1 measures, it also produces the weakest models when trained on only the Web blog corpus.

I can draw a few additional conclusions from these results. The Corman classifier has better performance when trained on the extremist corpus while the Gordon classifier has better performance with the web blog corpus. This is interesting because from the stories, the Extremist corpus mainly contains second hand accounts of events, often with a 3rd person narrator. On the other hand, the Web blog corpus mainly contains person stories and 1st person narrators. So it is possible that the Gordon classifier is better at finding personal stories while the Corman classifier is better at finding second hand accounts of events. It naturally follows that the Corman classifier has better performance when trained on the combined corpus than Gordon. This is due to the extremist corpus comprising 80% of the combined corpus.

5.6 Conclusions

I have introduced a new story detection approach which uses simple verb and character features. This new detector outperforms the prior state-of-the-art in all tasks, sometimes by orders of magnitude. Further, I showed that our detector generalizes significantly better across lexically different corpora. I propose that this increase in performance and generalizability is due to the more general nature of our features, especially those related to verb classes. This approach has additional advantages, for example, the feature vector is fixed in size and does not grow in an unbounded fashion as new texts (with new verbs, agents, and patents) are added to the training data.

In future work, I plan to develop richer character-based features. The current approach uses only normalized lengths of the five longest coreference chains, which leaves out important information about characters that could be useful to story detection. Indeed, our experiments showed that these character features only add a small amount of information above and beyond the verb features. However, when used alone, the character features still yield reasonable performance, which suggests that more meaningful character-based features could lead to story detectors with even better performance.

The applications of the story detector are numerous. Most concretely, I can point to its usage in the narrative level extractor presented in Chapter 4. Another application is for harvesting narrative text for further analysis.

Chapter 6

Automatic identification of narrative diegesis and POV[14]

The style of narrative news affects how it is interpreted and received by readers. Two key stylistic characteristics of narrative text are point of view and diegesis: respectively, whether the narrative recounts events personally or impersonally, and whether the narrator is involved in the events of the story. Although central to the interpretation and reception of news, and of narratives more generally, there has been no prior work on automatically identifying these two characteristics in text. I develop automatic classifiers for point of view and diegesis, and compare the performance of different feature sets for both. I built a gold-standard corpus where I double-annotated to substantial agreement ($\kappa > 0.59$) 270 English novels for point of view and diegesis. As might be expected, personal pronouns comprise the best features for point of view classification, achieving an average F_1 of 0.928. For diegesis, the best features were personal pronouns and the occurrences of first person pronouns in the argument of verbs, achieving an average F_1 of 0.898. I apply the classifier to nearly 40,000 news texts across five different corpora comprising multiple genres (including newswire, opinion, blog posts, and scientific press releases), and show that the point of view and diegesis correlates largely as expected with the nominal genre of the texts.

[14] This chapter was adapted from an article I published in the Proceedings of the Second Workshop on Computing News Storylines (Eisenberg, November 2016).

I released the training data and the classifier for use by the community. Additionally, I submitted a patent for the choice of what features to use for the automatic classification of narrative diegesis and POV. This patent was recently approved by the U.S. Patent Office (No. 15/804,589).

A year and a half after the development of the narrative characteristics classifiers, I realized that the POV and diegesis classifiers were invaluable tool for the task of narrative boundary extraction. In Chapter 4 I show how automatic classification of the POV and diegesis of paragraphs of text were useful in determining whether they belonged to embedded narratives. In certain experiments, the SVM could distinguish narrative levels based on change of POV between paragraphs. A key alteration was made to the POV and diegesis detectors from the pipeline discussed in this chapter: for narrative level detection do not remove quoted text. This is altered because I am interested in narrative characteristics of all texts, not just narrative speech, but also what characters say. Some embedded narratives are totally encapsulated by quotes, so I would miss the narrative characteristic changes present during these narratives.

6.1 Motivation

Interpreting a text's veridicality, correctly identifying the implications of its events, and properly delimiting the scope of its references are all challenging and important problems that are critical to achieving complete automatic understanding of news stories and, indeed, text generally. There has been significant progress on some of these problems for certain sorts of texts, for example, recognizing implications on short, impersonal, factual text in the long-running Recognizing Textual Entailment challenge (RTE[15]). On the other hand, narrative text (including much news writing) presents additional complications, in that to accomplish the tasks above one must take into account the narrator's point of view (i.e., first person or third person), as well as the narrator's personal involvement in the story (a feature that narratologists call diegesis).

In news stories specifically writers are encouraged to use the third person point of view when they wish to emphasize their objectivity regarding the news they are reporting (Davison, 1983). In opinion pieces or blog posts, on the other hand, first person is more common and implies a more personal (and perhaps more subjective) view (Aufderheide, 1997). News writers are also often in the position of reporting on events which they themselves have not directly observed, and in these cases can use an uninvolved style (known as hetereodiegetic narration) to communicate their relative remove from the action. When writers observe or participate in events directly, however, or are reporting on their own lives (such as in blog posts), they can use an involved narrative style (i.e., homodiegetic narration) to emphasize their personal knowledge and subjective, perhaps biased, orientation.

Before I can integrate knowledge of point of view (POV) or diegesis into text understanding, I must be able to identify them, but there are no systems which enable automatic classification of these features. In this paper I develop reliable classifiers for both POV and diegesis, apply the classifiers to texts drawn from five different news genres, demonstrate the accuracy of the classifiers on these news texts, and show that the POV and

[15] https://aclweb.org/aclwiki/Textual_Entailment_Portal

diegesis correlates much as expected with the genre. I release the classifiers and the training data so the field may build on our work and integrate these features into other text processing systems.

Regarding the point of view of the narrator, narratologist Mieke Bal claimed “The different relationships of the narrative ‘I’ to the objects of narration are constant within each narrative text. This means that one can immediately, already on the first page, see which is the [point of view].” (Bal, 2009, p. 29) This assertion inspired the development of the classifiers presented here: I had annotators mark narrative POV and diegesis from the first 60 lines of each of 270 English novels, which is a generous simulation of “the first page”. This observation allowed us to transform the collection of data for supervised machine learning from an unmanageable burden (i.e., having annotators read every novel from start to finish) into a tractable task (reading only the first page). I chose novels for training, instead of news texts themselves, because of the novels’ greater diversity of language and style.

Once I developed reliable classifiers trained and tested with this annotated data, I applied the classifiers to 39,653 news-related texts across five news genres, including: the Reuter’s corpus containing standard newswire reporting; a corpus of scientific press releases scraped from EurekAlerts; the CSC Islamist Extremist corpus containing ideological story telling, propaganda, and wartime press releases; a selection of opinion and editorial articles scraped from LexisNexis, the Spinn3r web blog corpus, and Reuters newswire. I checked a sample of the results, confirming that the classifiers performed highly accurately over these genres. The classifiers allowed us to quickly assess the POV and diegesis of the texts and show how expectations of objectivity or involvement differ across genres.

The chapter proceeds as follows. In §6.2 I define point of view and diegesis, and discuss their different attributes. In §6.3 I describe the annotation of the training and testing corpus, and then in §6.4 describe the development of the classifiers. In §6.5 I detail the results of applying the classifiers to the news texts. In §6 I outline related work, and in §6.7 I discuss how shortcomings of the work and how it might be improved. I summarize the contributions in §6.8. In short, this chapter asks the question: can point of view and diegesis be automatically classified? The experimental results in this chapter show that it can be done.

6.2 Definitions

6.2.1 Point of view

The point of view (POV) of a narrative is whether the narrator describes events in a personal or impersonal manner. There are, in theory, three possible points of view, corresponding to grammatical person: first, second, and third person. First person point of view involves a narrator referring to themself, and implies a direct, personal observation of events. In a third person narrative, by contrast, the narrator is outside the story's course of action, looking in. The narrator tells the reader what happens to the characters of the story without ever referring to the narrator's own thoughts or feelings. In theory second person POV is also possible, although exceedingly rare. In a second person narrative, the narrator tells the reader what he or she is feeling or doing, giving the impression that the narrator is speaking specifically to the reader themselves and perhaps even controlling their actions. This is a relatively rare point of view (in our training corpus of English novels it occurred only once), and because of this I exclude it from consideration. Knowing the point of view (first or third person) is important for understanding the implied veridicality as well as the scope of references within the text. Consider the following example:

(1) Donald made everyone feel bad. He is a jerk.

With regard to reference, if this is part of a first person narrative, the narrator is included in the scope of the pronoun everyone, implying that the narrator himself has been made to feel bad. In this case I might discount the objectivity of the second sentence if I know that the narrator himself feels bad on account of Donald. A third person narrator, by contrast, is excluded from the reference set, one can make no inference about his internal state and, thus, it does not affect our judgment of the implications of the accuracy or objectivity of later statements. With regard to veridicality, if the narration is third person, statements of fact can be taken at face value with a higher default assumption of truthfulness. A first person narrator, in contrast, is experiencing the events not from an external, objective point of view but from a personal point of view, and so assessment of the truth or accuracy of their statements is subject to the same questions as a second-hand report.

6.2.2 Diegesis

Diegesis is whether the narrator is involved (homodiegetic) or not involved (heterodiegetic) in the story. In a homodiegetic narrative, the narrator is not just the narrator but a character as well, performing actions that drive the plot forward. In a heterodiegetic narrative, the narrator is observing the action but not influencing its course. As reflected in Table 1, third person narrators are almost exclusively heterodiegetic, but first person narrators can be either. Like point of view, diegesis provides information to the reader on how to discount statements of fact, and so to judge the veridicality of the text.

6.3 Corpus

To train and test our classifiers I chose a corpus of diverse texts and had it annotated for point of view and diegesis. I used the Corpus of English Novels (De Smet, 2008), which contains 292 English novels published between 1881 and 1922, and was assembled to represent approximately a generation of writers from turn-of-the-century English literature. Novels were included in the corpus if they were available freely from Project Gutenberg (Hart, 2018) when the corpus was assembled in 2007. There are twenty-five authors represented in the corpus, including, for example, Arthur Conan Doyle, Edith Wharton, and Robert Louis Stevenson. Genres represented span a wide range including drama, fantasy, adventure, historical fiction, and romance. To simulate "the first page" of each novel, I manually trimmed each text file so that they started with the beginning of the first chapter. This was done by hand since automating this process was not a trivial task. Then, I automatically trimmed each file down to the first 60 lines, as defined by line breaks in the original files (which reflect the Gutenberg project's typesetting). These shortened texts were used by our annotators, and were the data on which the classifiers were trained and tested. I wrote an annotation guide for point of view and diegesis, and trained two undergraduate students to perform the annotations. The first 20 books from the corpus were used to train the annotators, and the remaining 272 texts were annotated by both annotators. After annotation was complete I realized that two of the files erroneously contained text from the preface instead of the first chapter, so I removed them from our study. Minus the training and removed texts, I produced a gold-standard corpus of 270 novels annotated for point of view and diegesis.

6.3.1 Inter-annotator agreement

I evaluated the inter-annotator agreement using Cohen's kappa coefficient (κ). For point of view κ was 0.635, which is considered substantial (Landis and Koch, 1977). The κ for diegesis is 0.592, almost substantial. Out of 270 markings, there were 36 and 33 conflicts between the annotators for POV and diegesis respectively. The first author resolved the conflicts in the POV and diegesis annotations by reading the text and determined the correct characteristic according to the annotation guide. I release this gold-standard corpus, including the annotation guide, for use by the community.[16]

[16] I have archived the code, annotated data, and annotation guide in the CSAIL Work Products section of the CSAIL Digital Archive, stored in the MIT DSpace online repository at https://dspace.mit.edu/handle/1721.1/29808.

6.3.2 Interaction of POV with diegesis

Table 1 shows the distribution of the texts in the corpus across the various categories. Of the 270 texts in the corpus, 74 had first person narrators, only 1 had second person, and 195 were third person. For diegesis, 55 were homodiegetic and 215 were heterodiegetic. There was only one second person narrator; this type of narrator is atypical in narrative texts in general, and I excluded this text from training and testing.

	First	Second	Third
Homodiegetic	54 (20%)	1 (0.4%)	-
Heterodiegetic	20 (7.4%)	-	195 (72.2%)

Table 6.1 Distribution of POV and diegesis

As I expected, there are no third person homdiegetic texts in the training corpus. Although in principle this is possible, it is narratively awkward, requires the narrator to be involved in the action of the story (homodiegetic), but report the events from a dispassionate, third-person point of view, never referring to themselves directly. Our data imply that this type of narrator is, at the very least, rare in turn of the century English literature. More generally, from our own incidental experience of narrative, I would expect this be quite rare across narrative in general.

6.4 Developing the classifiers

I implemented the preprocessing (§6.4.1), SVM training, cross-validation testing (§6.4.2), and feature extraction for the classifiers (§6.4.3 and §6.4.4) in Java (Gosling, 2014).

6.4.1 Preprocessing

The preprocessing was the same for both classifiers. The full text of the first 60 lines of the first chapter was loaded into a string, then all text within quotes was deleted using a regular expression. For both POV and diegesis it is important to focus on language that is uttered by the narrator, whereas quoted text represents words uttered by the characters of the narrative. The benefits of removing the quoted text is shown in Tables 3 and 4. After I removed the quoted text, I used the Stanford CoreNLP suite to tokenize and detect sentence boundaries (Manning et al., 2014). Finally, I removed all punctuation[17]. This produced an array of tokenized sentences, ready for feature extraction.

6.4.2 Experimental procedure

To determine the best sets of features for classification, I conducted two experiments, one each for POV and diegesis. In each case, texts were preprocessed as described above (§4.1), and various features were extracted as described below. Then I partitioned the corpus training and testing sets using ten-fold cross-validation. Precisely, this was done as follows:

[17] Specifically, the six characters [. ? ! , ; :].

for POV, the texts annotated as first person were divided into ten sets containing nearly equal numbers of texts, and I did the same for the third person texts. Then the first set of both the first person and third person texts were designated as the test sets and the classifier was trained on the remaining nine sets from each class. This was repeated with each set (second, third, fourth, etc. …), designating each set in order as the test set, with the remaining sets used for training. There are more third person narrators in the corpus; hence, each training fold has more examples of third person narrators than first person narrators. I performed cross-validation for diegesis in exactly the same manner.

I then trained an SVM classifier on the training fold using specific features as described below (Chang, 2011). To evaluate performance of the classifiers I report macro-averaged precision, recall, and F_1 measure. This is done by averaging, without any weighting, the precision, recall, and F_1 from each fold. I also report the average of F_1 for overall performance (weighted by number of texts).

6.4.3 Determining the best POV feature set

The best set of features for point of view should be straightforward: narrators either refers to themselves (first person) or they don't (third person). Naturally, a first-person narrator will refer to themself with first person pronouns, and so the presence of first person pronouns in non-quoted text should be a clear indicator of a first person point of view. Importantly, as soon as a narrator uses a first person pronoun they become a first person narrator, regardless of how long they were impersonally narrating. A list of the sets of first, second, and third person pronouns that I used as features can be found in Table 6.2.

I investigated eight different features sets for POV classification. The classifier with the best performance uses counts of the first, second, and third person pronouns as the feature set. Six of the remaining experiments use different subsets of the pronouns: I test the performance of on each individual set of pronoun as well as each combination of two pronouns sets. Features sets that did not consider first person pronouns were unable to classify first person narrations, but, importantly, first person pronouns alone were not the best for classifying first person narratives. The classifier that considers all three types of pronouns has an F_1 almost six percentage points higher than the classifier that only considers first person pronouns.

1st	I, me, my, mine, myself, we, us, our, ours
2nd	you, your, yours
3rd	he, him, his, she, her, hers, they, them, theirs

Table 6.2 Pronouns used for classification

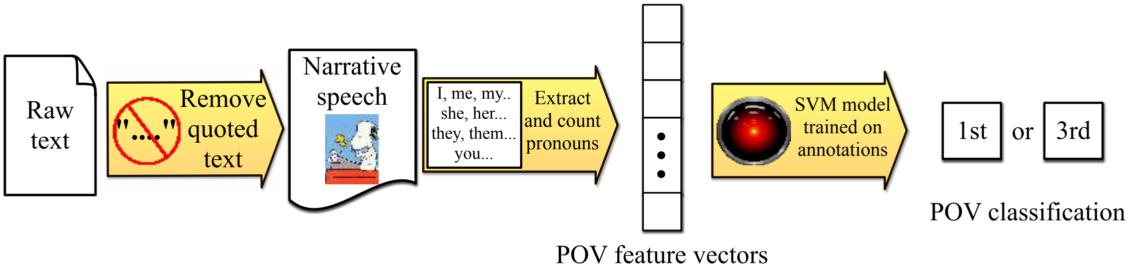


Figure 6.1 POV extraction pipeline

Previously I discussed that it is important to remove quoted text before the features are extracted. To test the importance, I ran an experiment where I did not remove quoted text in preprocessing, and then used all pronouns as in the best performing classifier. This negatively impacted F_1 for first person narrators by 13 percentage points and the F_1 for third person narrators by about 3 percentage points. This shows that it is important to remove quoted text before extracting features for POV classification. The only feature sets that did worse than the feature set with quoted text removed were those feature sets that did not include first person pronouns.

6.4.4 Determining the best diegesis feature set

Pronouns are also a prominent feature of diegesis, but it is not as simple as counting which pronouns are used: diegesis captures the relationship of the narrator to the story. On the one hand, if the narrator never refers to themself (i.e., a third person narrator), then it is extremely unlikely that they are participating in the story they are telling, and so they are, by default, a heterodiegetic narrator. On the other hand, first person narrators may be either homo- or heterodiegetic. In this case one cannot merely count the number and type of pronouns that occur, but must pay attention to when first person pronouns, which represent the narrator, are used as arguments of verbs that represent events in the story. Event detection is a difficult task (Verhagen, 2007), so I focus on finding when first person pronouns are used as arguments of any verb. While in reality not all verbs represent events, a large fraction do, and as the performance of the classifier shows this feature correlates well with the category. To find the arguments of verbs, I use our in-house semantic role labeler (SRL) that is integrated into the Story Workbench (Finlayson, 2008; Finlayson, 2011).

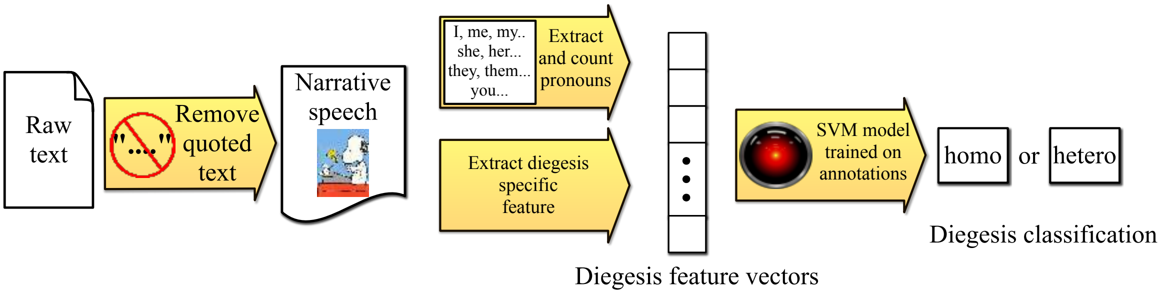

Raw text
Remove quoted text
"..."
Narrative speech
I, me, my.. she, her... they, them... you...
Extract and count pronouns
Extract diegesis specific feature
Diegesis feature vectors
SVM model trained on annotations
homo
or
hetero
Diegesis classification

Figure 6.2 Diegesis classification pipeline

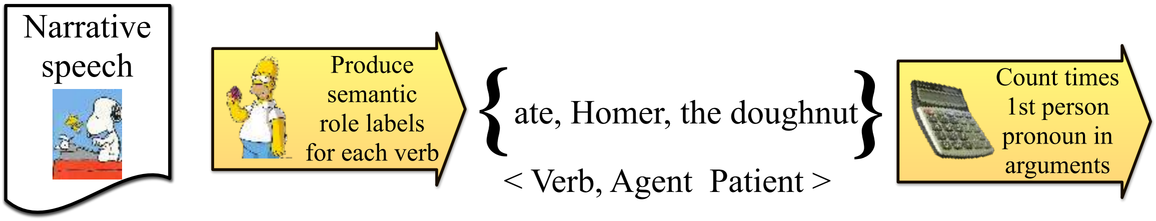

Narrative speech
Produce semantic role labels for each verb
{ate, Homer, the doughnut}
< Verb, Agent Patient >
Count times 1st person pronoun in arguments

Figure 6.3 1st person action extraction pipeline

I tested four different sets of features for diegesis classification. The simplest counts how many times each first person pronoun appears in an argument of a verb. Although this classifier is somewhat successful, it is somewhat weak identifying homodiegetic narrators.

The best performing diegesis classifier uses occurrences of the first, second, and third person pronouns in addition to the features from the simple diegesis classifier as features. I hypothesized that I could further improve the performance of this classifier by including a feature that counted the occurrences of second and third person pronouns as arguments of verbs that also have a first person pronoun as an argument (this is listed as the "cooccurrence" feature in Table 4). Our reasoning was that this feature would encode where the narrator and another character were connected by the same event, which is indicative of homodiegesis. Contrary to our expectations, however, this feature undermined homodiegetic classification: this classifier could not train an SVM model that could recognize homodiegetic narrators. This was the weakest of all of the diegesis classifiers.

Above I claimed that removal of quoted text is useful for diegesis classification. To show this, I took the feature set from our best diegesis classifier (with first person pronouns as arguments to a verb, and the occurrences of all pronouns), and took out the quoted text removal from the pipeline. This caused the F_1 measure to drop over 13 percentage points for homodiegetic and approximately 2 percentage points for heterodiegetic. These drops in performance indicate that the classifier performs better when quoted text is removed.

		First person			Third person			Avg
Feature Set	**Quoted text removed**	Precision	Recall	F1	Precision	Recall	F1	F1
Majority class baseline		0	0	0	0.72	1	0.84	0.61
3rd person pronouns only	X	0	0	0	0.73	9.99	0.84	0.61
2nd person pronouns only	X	0	0	0	0.75	0.98	0.85	0.62
2nd and 3rd person pronouns		0	0	0	0.74	0.98	0.84	0.61
All pronouns	X	0.91	0.67	0.74	0.89	0.96	0.92	0.874
1st person pronouns	X	0.97	0.7	0.79	0.9	0.99	0.94	0.9
1st and 3rd person pronouns	X	0.96	0.73	0.81	0.91	0.98	0.95	0.91
1st and 2nd person pronouns	X	0.94	0.76	0.81	0.92	0.97	0.94	0.91
All pronouns	X	0.94	0.81	**0.86**	0.94	0.97	**0.95**	**0.93**

Table 6.3 Results for POV classification experiments

			Homodiegetic			Heterodiegetic			Avg
Feature Set	**All pronouns**	**Quoted text removed**	Precision	Recall	F1	Precision	Recall	F1	F1
Majority class baseline			0	0	0	0.8	1	0.89	0.706
1st person pronoun as verb arg.	X		0.81	0.5	0.59	0.89	0.96	0.92	0.85
1st person as arg. + co-occurrence	X	X	0.85	0.48	0.59	0.89	0.98	0.93	0.86
1st person pronoun as verb arg.		X	0.91	0.58	0.68	0.91	0.98	0.94	0.89
1st person pronoun as verb arg.	X	X	0.93	0.62	**0.72**	0.91	0.98	**0.95**	**0.9**

Table 6.4 Results for diegesis classification experiments

6.5 Application of the classifiers to the news

To reveal the relationship of POV and diegesis to news story genres, I applied both classifiers a diverse set of news corpora. The classifiers for these experiments were trained on all 269 first and third person texts from the CEN,[18] using the best performing sets of features. I applied the classifiers to texts drawn from five corpora: the Reuters-21578 newswire corpus,[19] a corpus of scientific press releases scraped from EurekAlerts, a selection of opinion and editorial articles scraped from LexisNexis, the Spinn3r web blog corpus (Burton, 2009), and the CSC Islamist Extremist corpus containing ideological story telling, propaganda, and wartime press releases (Ceran, 2012).

The stories from the Spinn3r web blog corpus were found by Gordon and Swanson (2009) and the CSC Islamist extremist stories were found by Ceran (2012). These five corpora are used for testing the POV and diegesis classifiers; these corpora are not used for training the classifiers. For each experiment in this section, the best set of POV and diegesis features from S4.3 and §4.4, were used to train a classifier, these classifiers were trained on the first page of each novel from the CEN. For each corpora, after running the classifiers I randomly sampled texts and checked their classification to produce an estimate of the true accuracy of the classifiers. Sample sizes were determined by calculating the number of samples required to achieve a 99% confidence for a point estimate of proportion, using the proportion estimated by the classifier (Devore, 2011). In all cases the ratio of first person

[18] The number of texts was 269 because one text in the corpus of 270 texts was second person.

[19] http://www.daviddlewis.com/resources/testcollections/ reuters21578/

to third person texts (and homo- to hetero-diegetic texts) was chosen to be equal to the ratio in the classification.

Corpus	# Texts	1st Person	3rd Person	Homo.	Hetero.	Approx. accuracy
Reuters-21578	19,043	1 (<1%)	19,042 (~100%)	1 (<1%)	19,042 (~100%)	99% / 99%
EurekAlert	12,135	31 (<1%)	12,104 (~100%)	5 (<1%)	12,129 (~100%)	97% / 94%
CSC Extremist	3,300	42 (1%)	3,258 (99%)	15 (<1%)	3,285 (~100%)	94% / 92%
Lexis Nexis	4,974	1,290 (26%)	3,684 (74%)	818 (16%)	4,156 (84%)	70% / 40%
Spinn3r	201	133 (66%)	68 (34%)	67 (33%)	134 (67%)	42% / 21%

Table 6.5 Distribution of POV and diegesis across news and story corpora

6.5.1 Reuters-21578 newswire

This corpus contains 19,043 texts, and all but one were marked by the classifiers as third person and heterodiegetic. I expected this, as journalists typically use the third person POV and heterodiegetic narration to communicate objectivity.

The erroneous classification of one text as first person was the result of a type of language I did not anticipate. The article in question uses direct speech to quote a letter written by Paul Volcker, Federal Reserve Board chair, to President Ronald Reagan. The majority of the article is the text of the letter, where Volcker repeatedly refers to himself, using the pronoun “I”. The POV classifier interpreted this document at 1st person because the text of Volcker’s letter was not removed in the quotation removal phase. The letter is quoted using direct speech, which our simple, regular-expression-based quotation detection system cannot recognize.

To estimate the true accuracy of the POV classifier over the Reuters corpus I randomly sampled and checked the POV of 200 texts (including the single first person text). All of the classifications were correct except the single first person text, resulting in an accuracy estimate of 99.5% over the newswire text for the POV classifier (1.3% margin of error at 99% confidence).

To estimate the true accuracy of the diegesis classifier over this corpus I randomly sampled and checked the diegesis of 200 texts (including the single homodiegetic text). Of the 199 heterodiegetic texts, all were correct, while the single homodiegetic text was incorrect, resulting in an accuracy estimate of 99% for the diegesis classifier over the newswire text (1.81% margin of error at 99% confidence).

6.5.2 EurekAlert press releases

This corpus contains 12,135 texts scraped from EurekAlert,[20] dated between June 1st and December 31st, 2009. The distribution of this corpus is similar to the Reuters corpus, and over 99% of the texts were classified as third person and heterodiegetic narrations. Press offices write press releases to entice journalists to write newswire articles, and so it makes sense that they will attempt to mimic the desired narrative distance in the press release, seeking to present themselves as unbiased narrators.

To estimate the true accuracy of the POV classifier over the press releases I randomly sampled and checked the diegesis of 120 texts, including two first person and 118 third person. Of the two first person texts, one was correct, and of the 118 third person

[20] http://www.eurekalert.org

texts, 115 were correct, resulting in an accuracy estimate for the POV classifier of 97% over the press release text (4.03% margin of error at 99% confidence).

To estimate the true accuracy of the diegesis classifier over this corpus I randomly sampled and checked the diegesis of 120 texts, including 2 homodiegetic and 118 heterodiegetic. Of the two homodiegetic texts, neither were correct, and of the 118 heterodiegetic texts, 111 were correct, resulting in an accuracy estimate for the diegesis classifier of 94% over the press release text (5.6% margin of error at 99% confidence).

6.5.3 LexisNexis opinions and editorials

This corpus comprises 4,974 texts labeled opinion or editorial scraped from the LexisNexis website,[21] dated between January 2012 and August 2016. Texts were included if they contained more than 100 words and appeared in one of a set of major world publications including, for example, the New York Times, the Washington Post, and the Wall Street Journal. About one-quarter of these texts are first person, and more than half of the first person narrators were homodiegetic. I expected this increased abundance of first person and homodiegetic texts, as the purpose of these types of articles is often to express individual opinions or the writer’s personal experience of events.

To estimate the true accuracy of the POV classifier over the LexisNexis articles, I randomly sampled and checked the POV of 200 texts, 50 from those classified as first person and 150 from those classified as third person. Of the 50 texts classified as first

[21] http://www.lexisnexis.com/hottopics/lnacademic/

person all were confirmed correct, while of the 150 texts classified as third person only 90 were confirmed correct. This suggests that our classifier is not properly identifying all of the first person narrators in the LexisNexis corpus, and results in a accuracy estimate of 70% for the POV classifier over the LexisNexis texts (2.7% margin of error at 99% confidence).

To estimate the true accuracy of the diegesis classifier over this corpus I randomly sampled and checked the diegesis of 200 texts, including 24 homodiegetic and 126 heterodiegetic texts. Of the 24 homodiegetic texts, all were correct, and of the 126 heterodiegetic texts, 51 were correct, allowing us to estimate that the diegesis classifier has an accuracy of 40% over the press release text (11% margin of error at 99% confidence).

6.5.4 Spinn3r blogs

This corpus comprises 201 stories extracted by Gordon and Swanson (2009) from the Spinn3r 2009 Web Blog corpus (Burton, 2009). These texts come from web blogs, where people often tell personal stories from their perspective, or use the blog as a public journal of their daily life. In contrast with newswire text, there is no expectation that a blog will report the truth in an unbiased manner. The distribution of the POV on this corpus reflects this tendency, with 66% of the texts being first person.

The diegesis distribution for the web blog stories was not unexpected: slightly more than half of the blog stories with first person narrators are homodiegetic. These are the most personal stories of the web blog story corpus, in which the narrator is involved in the story’s action.

To estimate the true accuracy of the POV classifier on the Spinn3r corpus, I randomly sampled 20 texts, 13 from those classified as first person and 7 classified as third person. Of the 13 first person texts 9 were confirmed correct, while of the 7 third person texts only 3 were confirmed correct. Overall, our classifier has trouble classifying the web blog texts. This might be due to syntactic irregularities of blog posts, which vary in their degree of adherence to proper English grammar. With respect to third person narrators I estimate that the POV classifier has an accuracy of 42% over the web blog text (34% margin of error at 99% confidence).

To estimate the true accuracy of the diegesis classifier over this corpus I randomly sampled and checked the diegesis of 20 texts, including six homodiegetic and 14 heterodiegetic texts. Of the six homodiegetic texts, all were correct, and of the 14 heterodiegetic texts, three were correct. With respect to the heterodiegetic narrators I estimate that the diegesis classifier has an accuracy of 21% over the press release text (27% margin of error at 99% confidence).

6.5.5 Islamic extremist texts

The CSC Islamist Extremist corpus contained 3,300 story texts, as identified by Corman (2012). These texts were originally posted on Islamist Extremist websites or forums. Our POV classifier found that 99.7% of the extremist stories were written in the third person. For the most part, the extremist stories were second hand accounts of events, often to share news about the outcome of battles or recount the deeds of Jihadists.

To estimate the true accuracy of the POV classifier on this corpus, I randomly sampled 150 texts, 2 from those classified as first person, and 148 classified as third person. Both texts classified as first person were verified to be first person narrators. Of the 148 texts classified as third person, 139 were verified correct. With respect to third person narrators, I can estimate the classifier has an accuracy of 93.9% over the extremist texts (4.92% margin of error at 99% confidence).

To estimate the true accuracy of the diegesis classifier over this corpus I randomly sampled and checked the diegesis of 150 texts, including 2 homodiegetic and 148 heterodiegetic texts. Of the 2 homodiegetic texts, 1 was correct, and of the 148 heterodiegetic texts, 137. With respect to heterodiegtic narrators, I can estimate the classifier has an accuracy of 92% over the press release text (5.6% margin of error at 99% confidence).

6.6 Related work

As far as I know this is the first study on the automatic classification of point of view and diegesis at the level of the text. In his book "Computational Modeling of Narrative", Mani framed the problem of computational classification of narrative characteristics, including point of view and diegesis, defining with reference to narratology (Mani, 2012). He gives a framework for representing features and characteristics of narrative in his markup language NarrativeML. However, he does not actually implement a classifier for these characteristics.

Wiebe proposed an algorithm for classifying psychological point of view in third person fictional narratives (Wiebe, 1994). The algorithm is a complex rule-based classifier which tracks broadening and narrowing of POV, and reasons whether each sentence is objective or subjective. She discusses a study where people used the algorithm to classify sentences, but the accuracy of people in that task was not given. Thus, while intriguing, it is not clear how well this algorithm performs since its correctness was not verified with a human annotated corpus.

In more recent work, Sagae employed a data-driven approach for classifying spans of objective and subjective narrations (Sagae, 2013). Their experiments were performed on a corpus of 40 web blog posts from the Spinn3r 2009 web blog corpus (Burton, 2009). Their features included lexical, part of speech, and word/part of speech tag n-grams. The granularity of their classifier is fine grained, in that the system tags spans of text within a document, as opposed to our classifiers which classify the whole document.

6.7 Discussion

Our best classifier for POV uses the occurrence of all pronouns as features, with an F_1 of 0.857 for first person POV, and 0.954 for third person POV. The weighted average over the two classes is a 0.928 F1. Table 3 contains the results for the POV classification experiments. This is a great start for the automatic classification of POV, and comes close to human performance. It is reasonable and expected from narratological discussion that the best set of features is the number of first, second, and third person pronouns in non-quoted text.

The best diegesis classifier in our study, the one that counts the first-person pronouns as verb arguments as well as the occurrence of each pronoun, has an F_1 of 0.721 for homodiegetic, and 0.947 for heterodiegetic. The weighted average over the two classes is a 0.898 F_1. Table 6.4 contains the results for the diegesis classification experiments. This is a good first start for diegesis classification, but the performance for homodiegetic narrators falls short. The features for this classifier are also reasonable: first person pronouns in verb arguments shows that the narrator is either causing action to happen or being affected by actions, and so should naturally correlate with homodiegesis. The inclusion of all pronouns as a feature for diegesis also makes sense, as point of view and diegesis are closely correlated. As noted previously, third person narrators cannot refer to themselves, so they cannot be related to the story.

The best performing POV and diegesis classifiers performed significantly than their respective baseline classifiers. In Table 6.3, the majority class baseline classifier has 0.607 F_1, while the best POV classifier has 0.928 F_1. Table 6.4 shows that the majority baseline classifier for diegesis has 0.706 F_1, while the best diegesis classifier has 0.898 F_1.

Diegesis classification might be improved by restricting pronoun argument detection only to those verbs that actually indicate events in the story. This focuses the classifier on places where the narrator is involved in driving the story forward, which is more closely aligned with the definition of diegesis. To do this, I would need to incorporate an automatic event detector (Verhagen et al., 2007, e.g.). On the other hand, event detection currently is not especially accurate, and incorporating such a feature may very well depress our classification performance.

Another approach of interest would be to adapt our classifiers to detect how a narrative characteristic changes over the course of a text. Our study focused on short spans of traditional, formal, edited novels where the point of view and diegesis remained constant. In longer texts it is possible that these characteristics could change, for example, in a stream of text comprised of multiple narratives, or in a text which explicitly is trying to defy convention (e.g., in highly literary texts such as James Joyce's Ulysses).

Finally, our classifier assumed that the classified texts were all approximately the same length (i.e., the first page, or approximately 60 lines). A modification that would be important to explore is using densities or ratios for the occurrences of the pronouns, instead of raw counts, for classifying texts that are less than 60 lines long.

6.8 Contributions

In this paper, I described and made significant progress against the problem of automatic classification of narrative point of view and diegesis. I demonstrated a high performing classifier for point of view with 0.928 F_1, and a good classifier for diegesis with 0.898 F_1. To evaluate our classifiers I created a doubly annotated corpus with gold-standard annotations for point of view and diegesis– based on the first 60 lines–of 270 English novels. I applied these classifiers to almost 40,000 news story texts drawn from five different corpora, and show that the classifiers remain highly accurate and that the proportions of POV and diegesis they identify correlates in an expected way with the genre of the news texts. I provide the annotation guide, annotated corpus, and the software as resources for the community.

Important to rehash, the POV and diegesis classifiers were very useful in the task of narrative boundary extraction. This is a first real world task that can now be solved by computers, using the insights gained via the training of the models in these experiments.

Chapter 7

Related work

My research is informed and enhanced by the research of others. My work would not be possible if it wasn't for prior efforts from other researchers. In this chapter I will discuss work that is similar to mine. First, in §7.1, I will discuss previous approaches to computational understanding of narrative structure. Second, I will show how my programs, specifically my NLP pipelines, use other researchers' programs as black boxes. In §7.2 I will discuss other programs that I used in my pipelines. Some of the most frequently used programs were Stanford CoreNLP for preprocessing (Manning, 2014) and coreference resolution (Clark, 2016), It Makes Sense's word sense disambiguation (Zhong, 2010), and the semantic role labeler from Story Workbench (Finlayson, 2008 & 2011).

Additionally, in §6.55 I already presented prior work on the task of story classification. Here I explained the Gordon (2009) and Corman (2012) story classifiers, and how the results of their work influenced me to build my story detector §6.3.

7.1 Computational models of narrative

7.1.1 Learning Proppian functions

My advisor, Mark Finlayson, worked on extracting high-level narrative structure from stories, specifically Russian folktales (Finlayson, 2016). Mark built programs that can automatically extract Proppian functions from Russian folktales. A Proppian function is “an act of a character, defined from the point of view of its significance for the course of the action” (Propp, 1968, p.21). Some common functions are, an initial act of villainy, struggle/victory, and reward for success. This work is the “...first demonstration of a computational system learning a real theory of narrative” (Finlayson, 2016).

For this work, Mark ran a semantic annotation study, for “semantic roles, coreference, temporal structure, event sentiment analysis, and dramatis personae” on 15 Russian folktales. These annotations were used to automatically extract Proppian functions from folktales. This work helped inspire my research. His work is on extracting the internal structure of stories, specifically trying to group stories with similar events together, while my work is on trying to understand how a story is told (where does the story begin, is there narrative text, how does the narrator tell the story). My work can be used to filter raw text for narrative text, and then Mark’s programs can be used to understand the fine-grained semantics of the story.

7.1.2 Narrative event chains, narrative cloze, and story cloze

An interesting, and enjoyed, task for NLP researchers is “narrative cloze”. For this problem computers are usually challenged to the following: Given a paragraph of narrative text, randomly remove a sentence. Can the computer automatically predict the verb and its arguments for the missing sentence? In 2008, Chambers and Jurafsky created this task to be “…a comparative measure to evaluate narrative knowledge” (Chambers, 2008). It’s an artificial task, since it is hard to say whether humans can consistently solve it, but NLP researchers often participate in shared tasks where the goal is to solve this problem. Also in 2008, Chambers defined narrative event chains as a "…partially ordered set of events that share a common actor (Chambers 2008)."

In 2016 Mostafazadeh introduced the Story Cloze task (Mostafazadeh, 2016). It is similar to the narrative cloze task, but with a bit more common sense: given a four-sentence story, can the computer automatically guess what the final sentence of the story is? This is a more natural task than narrative cloze; people try to guess the last event all the time; think about people trying to predict what happens at the end of a TV show or movie. To obtain data for this task, Mostafazadeh paid workers from Amazon Mechanical Turk[22] to create a corpus of 50,000 five sentence stories. These stories, and about 48,000 other five sentence stories can be downloaded from the "Story Cloze Test and ROC Stories Corpora" website[23]. The goals of this collected data were to 1) "enable learning of narrative structure across a range of events as opposed to a single domain or genre" and 2) "to train rich coherent story-telling models" (Mostafazadeh, 2016).

In 2017, Mostafazadeh used his corpus of five sentence stories for a story cloze shared task (Mostafazadeh, 2017). In this task, the two best submissions had accuracy between 0.74 and 0.76. Both systems used logistic regression models. The best models use features that model the language of concluding sentences, and does not use information from the previous sentences. The second-best system has a more natural set of features: coherence of events throughout the proposed story, "emotional trajectory", and "plot

[22] https://www.mturk.com/

[23] http://cs.rochester.edu/nlp/rocstories/

consistency". These systems did better than a neural network based baseline, with an accuracy of 0.60.

7.1.3 Event extraction

Understanding events, is essential for understanding stories. Plot cannot be extracted without knowing what events are portrayed in a text. Therefore, accurate extraction of events is important for computational modeling of narrative. Event extraction is hard because not every verb represents an event. Some verbs are used to tell the narratee details, not everything is plot. A notable work in event extraction was carried out by Chambers (2014). He has open sourced a program called CAEVO, which can automatically extract events, and construct a graph that explicitly orders every pair of extracted events ("dense event graphs"). The system has a 0.51 F_1, for the task of determining two events' temporal relationship (i.e. before, after, includes, is included, simultaneous, vague), which is a "14% relative increase over the top two systems in TempEval3".

7.1.4 Temporal relationship of events

Some work has been done on determining the temporal relationship between two events in a story: does one event precede the other event, or are they simultaneous? A recent work, by Yao (2018), improved the state of the art on several tasks in temporal relationship classification. To train their classifier, Yao needed a large corpus of narrative text. Yao obtained a narrative corpus by using a variation of my story detector (Eisenberg, 2017).

They use a maximum entropy classifier, with a rule based filtering phase. Their classifier requires that every paragraph must have a “protagonist” to be classified as a narrative paragraph. To enforce this rule, they run a test on the longest coreference chain, they normalize its length by the number of sentences in the paragraph. Finally, the paragraph can only move onto the next phase—a maximum entropy classifier—if the normalized length of the longest coreference-chain is greater than or equal to 0.4. The maximum entropy phase uses features based on occurrence of characters, plot events, part of speech tags, words that “…denote relativity (e.g., motion, time, space) and words referring to psychological processes (e.g., emotion and cognitive)”. This is an adaptation of my character-based feature discussed in §5.3.2.

They used their story extractor to filter three corpora: Novels from the BookCorpus (Zhu, 2017), news articles from the English Gigaword, 5th edition (Graff, 2003), and blog posts in the Blog Authorship Corpus (Schler, 2006). Their classifier was used to find 237,000 paragraphs of text from these corpora. Then these paragraphs of text were used to train a classifier in extracting temporal relationships between events.

7.2 NLP tools

A huge resource to the implementation of my programs, was my ability to use open source tools. A Ph.D. worth of research could have been spent developing each one of the tools that I use in my NLP pipelines. I treat these tools as *black boxes*, and assume they are accurate tools. In fact, each of these tools are complicated, and their careful development has enabled the success of my classifiers and extractors. In this section I will briefly cover the tools that I used, and how they allowed me to accurately teach computers to understand narrative structure.

7.2.1 Stanford CoreNLP

The Stanford CoreNLP Java library was used in every one of my NLP pipelines (Manning 2014). Stanford CoreNLP is a Java library which provides functionality that is necessary for almost any NLP pipeline:

- Tokenization - breaking streams of text into lists of words
- Named entity recognition - finding which spans of text represent people, organizations, and time-phrases
- Coreference resolution chain extraction - making lists of references to each referent in a text
- Part-of-speech tagging - determining which of the 36 Penn Treebank POS (Marcus, 1993).

7.2.2 Story Workbench semantic role labeler

While working on his Ph.D., my advisor, Mark Finlayson wrote a semantic role labeler (SRL), and integrated it into a larger tool, the Story Workbench (Finlayson, 2008 & 2011). SRL is a tool, given a parse of a sentence, that can identify the arguments of each predicate. This tool was useful for determining who is participating in the actions that are talked about in a text. Remember, events can be expressed in language with verbs. The agent and patient argument can tell us who (or what) is causing the event, and who the event is being acted upon. When the arguments of verbs contain characters, or pronouns which represent characters, this is a signal that a span of text contains narrative content. When the arguments of a verb contain first person pronouns, this is often a signal of a homodiegetic narrator.

There are other SRLs (Zhou, 2015) that have better performance than the Story Workbench SRL, but in 2015 and 2016, when I was implementing the Corman story classifier, and my diegesis detector, the time investment for getting other SRL systems up and running was too great a hurdle to jump while maintaining the deadlines I needed to meet for my Ph.D. research to progress.

7.2.3 It Makes Sense word sense disambiguation

Word sense disambiguation (WSD) is the process of determining the meaning of each word in a text. The letters that spell out a word can have different meanings depending on the surrounding context. For example, consider the two sentences:

(1) *Duck* is my favorite bird to eat.

(2) The ball was coming towards me, so I had to *duck.*

The word *duck* appears in both sentences, but they don't mean the same thing. In sentence (1) the sense that refers to the type of bird that swims in water is used, but in sentence (2) duck refers to the action of lowering your head to avoid being hit. WSD programs give computers the ability to take raw text and determine the word sense for each word.

For my story detector, I used the *It Makes Sense (IMS) WSD* (Zhong, 2010) Java Library. IMS is one of the first open source WSD tools for Java, and it was state of the art when it was first released. It was trained and evaluated on SensEval and SemEval data labeled data, and it uses linear SVM classifiers.

7.2.4 VerbNet, the Java VerbNet Interface, and event extraction

VerbNet is a lexicon for verbs (Schuler, 2005). This lexicon includes a hierarchically organized set of verb classes, which groups semantically similar classes of verbs together (like verbs relating to making music are all grouped together in one class). In my pipelines, I interact with VerbNet using the Java VerbNet Interface (Finlayson, 2012), a JavaLibrary which allows for API access to the verb lexicon. Specifically, I use JWI and VerbNet in my Story Classifier (Chapter 5) for extracting verb vectors from raw text. First I use *It Makes Sense WSD* (Zhong, 2010) to get the sense tag for each verb, and then I use JWI to access VerbNet for the class that the verb belongs to.

Computational understanding of verbs is important because verbs in text can represent the occurrence of events and action. Sometimes events are represented by spans of text longer than just a single verb. Sometimes events are represented by nouns. Event extraction is a tough problem in NLP, however there are some automatic event extraction programs, most notably CAEVO (Cascading Event Ordering architecture) (Chambers, 2014), which extracts events from raw text and arranges them in a time series. Although this system has state of the art performance, I was unable to get it working. It would have been useful to use CAEVO since I would be able to analyze all the events in a text, and not just the verbs. In my current system, I only analyze the verbs, which is limiting, since not all verbs represent events, and sometimes there are events talked about without using verbs. Eventually, it would be beneficial for my programs, that automatically understand narrative structure, to get an event extractor working.

Chapter 8

Conclusion

In conclusion, my work has taught computers to extract narrative structure from text. Specifically, my work has enabled computers to automatically extract narrative POV and diegesis, decide which paragraphs of text have story content, and extract narrative levels from long texts. Extraction of POV, diegesis, and narrative levels are all abilities that computers did not have until my work. These key features have enabled computers to have a better awarenesses and understanding of stories and narrative that are second nature to humans. I am not claiming that I have taught computers everything they need to know about understanding narrative, but my work has created certain structural elements of narrative (POV, diegesis, and narrative levels) that computers can now classify because of my research. In the rest of this chapter, I will cover some next steps that expand upon the research in my dissertation (§8.1). Then, I will discuss how my work can be applied to the domain of automatic stock trading (§8.2).

8.1 Future work

8.1.1 Narrative level extractor

I have two recommended improvements for the narrative level extractor. First, the narrative level extractor only extracts the first level of embedding or interruption. It cannot yet decipher if there are multiple levels of embedding. It can only detect if there is a shift from the original narrative to a new narrative. The narrative level extractor can be modified in the following way to account for multiple levels of embedding or interruption: Once the detector finds a new narrative, it records which sentences belong to the new level. Then run the detector again on text where each sentence from the new narrative are appended to each other. This recursive process should be repeated for each new narrative level that is found by the detector. I hypothesize that the main character features will be most useful for finding these doubly embedded narratives, because the set of characters, especially the protagonist, tends to change across narrative levels.

The second enhancement would be teaching the computer to cluster the sentences from embedded or interruptive narrative levels, into distinct narratives. Currently the extractor is only aware that there are new narrative levels. The extractor cannot decide which spans belong to which narrative level. Ideally, the computer should know how many narrative levels are in a text, and which sentences belong to which level. I surmise that this can be accomplished, automatically, by using clustering algorithms and topic modeling. Additionally, analysis of which characters are mentioned in which spans should be helpful for making this decision, since each narrative level tends to have different sets of characters.

8.1.2 Story detector

Currently the story extractor only works accurately for texts that are one paragraph long. Sometimes, it is important to know whether a sentence in a paragraph contains any story content. A whole story will usually not appear in a single sentence, but a single sentence could have rich character information, and details about an event as it is unfolding. Adapting the story extractor to run on sentences, will allow people to get classifications for each sentence, which is useful for tasks where knowing info about single sentences is necessary.

For my narrative level extractor (Chapter 4), I use the story extractor (Chapter 5) to produce story classifications for each paragraph. Then I propagated these classifications down to each sentence in the paragraph. It would have been more accurate to classify each sentence, instead of assuming the *storiness* of the sentence is the same as its paragraph.

It would be useful to obtain more annotated data for the task of story detection. The two data sets that are currently available are highly unbalanced. Less than 20% of the annotated data contains positive examples of stories. Also, it is unnatural to annotate paragraphs for story content. Stories can begin or end in the middle of paragraphs, which is like narrative levels. Also, stories can be interrupted at any point in time. It's worth rethinking the paradigm of annotating texts with paragraph granularity for a binary story annotation, and consider annotating longer passages of text, for spans that contain story content.

8.1.3 POV and diegesis extractor

Similar to my ideas about improving the story detector, the POV and diegesis detectors need to be augmented to classify short spans of text, specifically for single sentences. Currently, the POV and diegesis detectors can only classify full paragraphs. This restriction affected the performance of the narrative level extractor, because the classification for each paragraph was propagated down to the sentence level. This procedure does not always produce correct annotations at the sentence level.

Additionally, it would be useful to run a new annotation study for POV and diegesis, where annotators are working on long passages of text, and they annotate the POV and diegesis for each span. This is important because either of these characteristics can change at any point in a text, not just in between paragraphs.

There is also room for improvement in the actual classes the POV extractor can classify. Currently, it classifies everything as first or third person POV. In narratology, second person POV is a real type of POV. I didn't include second person POV in my experiments because my annotators didn't find enough second person narrators to allow for training and testing a SVM model. It would be interesting to run a new annotation study that is specifically aimed at finding more second person narrators, to enable the SVM to learn how to classify this type of POV. Further, considering POV more closely, as personal pronouns have become more fluid and specific to identify gender-nonconforming people, it would be important to consider the evolution and use of LGBTQ+ pronouns in storytelling as research in this field develops.

8.2 Automatic stock trading

One of the most financially appealing applications of my work is using narrative structure extraction along with other NLP methods to guide automatic stock trading. Hedge funds already use NLP to analyze social media, and newswire to decide when to buy or sell stocks. They use primitive methods, like sentiment analysis and named entity recognition to make decisions. The following is a simplistic example: An article is published on a financial news website. A computer at a hedge fund will process an article, and use it to make decisions about what to do with its portfolio of investments. If the article mentions "Amazon" and its CEO "Jeff Bezos" enough times, the computer will know that the article is about the company Amazon. If the article has a positive sentiment, then the computer will buy or hold onto Amazon's stock, because something "positive" is happening at Amazon. If the article has a negative sentiment, then the computer will sell stock, or short the stock, because it is receiving "negative" press. These are the types of programs that are currently being used to analyze newswire and social media to make decisions about how to play the stock market. While these programs enable companies to make quick decisions automatically, the decisions are not necessarily intelligent or based in understanding the context of the data being used.

Deciding how to trade stocks is more complicated than knowing how positive or negative an article is. It is more important to understand what events are being mentioned, and predicting the impact these events will have: Can the events in this article be matched with events in the past to help make an educated decision? Also, it is important to know what parts of an article (or social media post) are referring to past, current or future events. Sometimes articles will give a history of a company, which accompanies reporting of a current event. Sometimes articles are written in a personal manner, and give more opinions than facts.

The processes for narrative structure extraction that I developed can be used to gain a more nuanced computational understanding of stories than sentiment analysis. First, POV and diegesis extraction can be used to decide whether a text is written in a personal or biased manner, or if it is more objective reporting. First person narratives tend to give their opinions, and not just report facts, while third person narrators are telling a story without making it personal, so they tend to be more objective. Knowledge of the POV and diegesis of a text can be used to determine how objective it is.

Second, narrative level extraction can be used to parse through different narratives in an article. Usually, the beginning of the article will contain a telling of the events that just happened, and interspersed throughout the remainder of the text will be narratives about the past. It is important to process the many narrative levels separately. Typically, the most important story to process is the one about the most recent events. The stories about the past should communicate information that the computer already knows. My narrative level extractor can be used to let the computer know that there are distinct narratives being told. Then, other programs can be used to determine which narrative is about the most recent events, and then these spans of text should be used to make decisions about what to do in the stock market.

Finally, once narrative levels have been extracted from an article, more fine-grained information can be extracted. Namely, what events are mentioned in each narrative. Once the computer knows what events come from the most current narrative, it can try to make decisions based on the sequence of events. Techniques like Analogical Story Merging (Finlayson, 2016) can be used to cluster the events in the narrative with sequences of events from the past. Once the computer can classify the current event sequence with sequences from the past, it will be easier to predict how the current events will affect stock performance, and enable better decision making.

REFERENCES

Abbott, H.P. (2008) *The Cambridge Introduction to Narrative*. Cambridge, UK: Cambridge University Press.

Apache Foundation. (2017) *Welcome to Apache OpenNLP.* Retrieved from https://opennlp.apache.org/

Atwood, M. (2000) *The Blind Assassin.* Toronto, Ontario: McClelland & Stewart.

Atwood, M. (2017) *The Handmaid's Tale*. Boston, MA : Houghton Mifflin Harcourt.

Aufderheide, P. (1997) *Public Intimacy: The Development of First-Person Documentary.* In Afterimage: The Journal of Media Arts and Cultural Criticism (Vol. 25, No. 1, pp 16-18).

Bal, M. (2009) *Narratology: Introduction to the theory of narrative.* Toronto, Ontario: University of Toronto Press.

Braud, C., & Denis, P. (2014*). Combining Natural and Artificial Examples to Improve Implicit Discourse Relation Identification.* In Proceedings of the 25th International Conference on Computational Linguistics (COLING 2015, pp. 1694-1705).

Bruner, J. (1991). *The Narrative Construction of Reality*. In Critical Inquiry (Vol. 18, No. 1, pp 1-21).

Burton, K., Java, A., & Soboroff, I. (2009) *The ICWSM 2009 Spinn3r Dataset*. In Proceedings of the 3rd Annual Conference on Weblogs and Social Media (ICWSM 2009).

Bush v. Gore. 531 U.S. 98 (2000).

Ceran, B., Karad, R., Corman, S., & Davulcu, H. (2012)*. A Hybrid Model and Memory Based Story Classifier.* In Proceedings of the 3rd Workshop on Computational Models of Narrative (CMN, pp. 58-62).

Chambers, N., Jurafsky, D. (2008). *Unsupervised Learning of Narrative Event Chains*. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-08, pp. 789-797).

Chambers, N., Cassidy, T., McDowell, B., & Berthard, S. (2014). *Dense Event Ordering with a Multi-Pass Architecture*. In Transactions of the Association for Computational Linguistics (Vol. 2, pp. 273-284).

Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (*TIST, Vol. 2, No. 3, Article 27).

Charles, L. (Writer), & David, L., & Seinfeld, J. (Directors). (1993, October 13) The Bris. [Television series episode] In David, L. (Executive Producer), *Seinfeld.* New York, NY: NBC Television Distribution.

Conrad, J. (2016) *Heart of Darkness*. New York, NY: W. W. Norton & Company.

Clark, K., & Manning, C. D. (2016). Improving coreference resolution by learning entity-level distributed representations. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers, pp. 643-653).

Clarke, J., Srikummar, V., Sammons, M., Roth, D. (2012). An NLP Curator (or: How I Hearned to Stop Worrying and Love NLP Pipelines). In Proceedings of the International Conference on Language Resources and Evaluation (LREC'12, pp. 573-580).

Dancygier, B. (2012) *The Language of Stories: A Cognitive Approach*. Edinburgh, UK: Cambridge University Press.

Davison, W.P. (1983). *The third-person effect in communication.* In Public Opinion Quarterly (Vol. 47, No. 1, pp. 1-15).

De Smet, H. (2008) *The Corpus of English Novels.* Retrieved from https://perswww.kuleuven.be/~u0044428/cen.htm

Devore, J.L. (2011). *Probability and Statistics for Engineering and the Sciences.* Boston, MA: Engage Learning.

Dredze, M., Crammer, K., Pereira, F. (2008). *Confidence-Weighted Linear Classification.* In Proceedings of 25th International Conference on Machine Learning (ICML'08, pp 264-271).

Eisenberg, J. D., Yarlott, W. V. H., & Finlayson, M.A. (July 2016). *Comparing Extant Story Classifiers: Results & New Directions*. In Proceedings of the 7th International Workshop on Computational Models of Narrative (CMN, paper 6).

Eisenberg, J.D., & Finlayson, M.A. (November 2016). *Automatic Identification of Narrative Diegesis and Point of View.* In Proceedings of the Second Workshop on Computing News Storylines (CNewsStory 2016, pp. 36-46)

Eisenberg, J.D., & Finlayson, M.A. (2017). *A Simpler and More Generalizable Story Detector using Verb and Character Features.* In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2017, pp. 2708-2715).

Eisenberg, J.D. & Finlayson, M.A. (2017). *U.S. Patent No. 15/804,589.* Washington, DC: U.S. Patent and Trademark Office.

Eisenberg, J.D. & Finlayson, M.A. (2018). *U.S. Patent Application No. 62/728,380.* Washington, DC: U.S. Patent and Trademark Office.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Finkel, J.R., Grenager, T., Manning, C. (2005) *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling.* In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005, pp. 362-370).

Finlayson, M. A. (2008). *Collecting Semantics in the Wild: The Story Workbench*. In Proceedings of the AAAI Fall Symposium: Naturally-Inspired Artificial Intelligence (pp. 46-53).

Finlayson, M.A. (2011). *The Story Workbench: An Extensible Semi-Automatic Text Annotation Tool.* In Proceedings of the 4th Workshop on Intelligent Narrative Technologies (INT4, pp. 21-24).

Finlayson, M.A. (2012). *JVerbnet, v 1.2.0,* Retrieved from http://projects.csail.mit.edu/jverbnet

Finlayson, M.A. (2014). *Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation.* In Proceedings of the 7th International Global Wordnet Conference (GWC 2014, pp 78-85).

Finlayson, M.A. (2016). *Inferring Propp's Function from Semantically Annotated Text.* In the Journal of American Folklore (Vol. 129, No. 511, pp. 55-77).

Forster, E.M. (2010) *Aspects of the Novel.* New York, NY: Rosetta Books.

Gilligan, V. (Writer), & Bole, C. (Director). (1998, February 22) Bad Blood. [Television series episode] In Carter, C. (Executive Producer), *The X-Files.* Los Angeles, CA: FOX Broadcasting Company.

Gordon, A., & Swanson, R. (2009, May). Identifying personal stories in millions of weblog entries. In Third International Conference on Weblogs and Social Media, Data Challenge Workshop, San Jose, CA (Vol. 46).

Gosling, J., Joy, B., Steele, G., Bracha, G., & Buckley, A. (2014). *The Java Language Specification, Java SE 8 Edition (Java Series).* Boston, MA: Addison Wesley Professional.

Graff, D. & Cieri, C. (2003). *English Gigaword Corpus.* Edition 5. https://catalog.ldc.upenn.edu/LDC2003T05

Grand Jury Testimony of Monica S. Lewinsky: Hearings before the Judiciary Committee, House of Representatives, 106th Cong. (2000).

Hart, M. (2018). *Free ebooks – Project Gutenberg.* Retrieved from https://www.gutenberg.org/

Haven, K. (2007). *Story Proof: The Science Behind the Startling Power of Story.* Westport, CT: Libraries Unlimited.

Jansen, P., Surdeanu, M., & Clark, P. (2014). *Discourse Complements Lexical Semantics for Non-factoid Answer Reranking.* In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014, pp. 977-986).

Japkowicz, N. (2000). *Learning from imbalanced data sets: a comparison of various strategies.* In AAAI workshop on learning from imbalanced data sets (Vol. 68, pp. 10-15).

July, M. (2015) *The First Bad Man: A Novel*. New York, NY: Scribner.

Keerthi, S.S., Lin, C.J. (2003). *Asymptotic Behaviors of Support Vector Machines with Gaussian*
Kernel. In Neural Computation (Vol. 15, No. 7, pp. 1667-1689).

Kingsbury, P., Palmer, M. (2003). *Propbank: the Next Level of Treebank.* In Proceedings of
Treebanks and Lexical Theories (Vol. 3).

Landis, J.R., Koch, G.G. (1977) *The Measurement of Observer Agreement for Categorical Data*.
In Biometrics (Vol. 33, pp. 159-174).

Mani, I. (2012) *Computational Modeling of Narrative.* Williston, VT: Morgan & Claypool
Publishers.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). *The*
Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 55-60).

Marcus, M.P., Marcinkiewicz, M.A., & Santorini, B. (1993) *Building a Large Annotated Corpus*

of English: The Penn Treebank. In Computational Linguistics (Vol. 19, No. 2, pp. 313-330).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Advances in Neural Information Processing Systems (pp. 3111-3119).

Moore, R., & Braga, B. (Writers), & Kolbe, R. (Director). (1994, May 23) All Good Things…. [Television series episode] In Berman, R. (Executive Producer), *Star Trek: The Next Generation.* Santa Monica, CA: CBS Television Distribution.

Moore, R. (Writer), & Landau, L. (Director). (1993, February 15) Tapestry. [Television series episode] In Berman, R. (Executive Producer), *Star Trek: The Next Generation*. Santa Monica, CA: CBS Television Distribution.

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., & Allen, J. (2016). *A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories.* In Proceedings of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies (NAACL-HLT) (pp. 839-849).

Mostafazadeh, N., Roth, M., Louis, A., Chambers, N.. & James, A.F. (2017). *LSDSem 2017 Shared Task: The Story Cloze Test.* In the 2017 Proceedings of the 2nd Workshop on Linking Models of Lexical, Semantic and Discourse-level Semantic (LSDSem 2017, pp. 46-51).

Murakami, H. (2011) *1Q84*. New York, NY: Vintage International.

Murakami, H. (2014) *Colorless Tsukuru Tazaki and his years of pilgrimage: a novel*. New York, NY: Alfred A. Knopf.

Nelles, W. (1997) *Frameworks: Narrative Levels & Embedded Narrative*. New York, NY: Peter Lang Publishing.

Ng, H.T., Goh, W.B., Low, K.L. (1997). *Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization.* In Association for Computing Machinery's Special Interest Group on Information Retrieval Forum (Vol. 21, pp 67-73).

Nissim, M., Abzianidze, L., Evang, K., van der Doot, R., Haagsma, H., Plank, B., Wieling, M. (2017). *Sharing is caring: The Future is Shared Tasks.* In Computational Linguistics (Vol. 43, No. 4, pp. 897-904.

Obergefell v. Hodges. 576 U.S. __ (2015).

Punyakanok, V., Roth, D., Yih, W.T. (2008). *The Importance of Syntactic Parsing and Inference in Semantic Role Labeling*. In Computational Linguistics (Vol. 34, No. 2, pp. 257-287).

Propp, V. (1968). *Morphology of the Folktale.* Austin, TX: University of Texas Press.

Ratinov, L., Roth, D. (2009). *Design Challenges and Misconceptions in Named Entity Recognition.* In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (pp. 147-155).

Rosenblatt, F. (1958). *The Perceptron: a Problabilistic Model for Information Storage and Organization in the Brain.* In Psychological Review (Vol. 65, No. 6, pp. 386-408).

Rowling, J.K. (2000). *Harry Potter and the Goblet of Fire*. New York, NY: Scholastic Press.

Sagae, K., Gordon, A. S., Dehghani, M., Metke, M., Kim, J. S., Gimbel, S. I., & Immordino-Yang, M. H. (2013). *A data-driven approach for classification of subjectivity in personal narratives.* In *Proceedings of the 5th International Workshop on Computational Models of Narrative (CMN'13 pp. 198-213).*

Salton, G., Wong, A., Yang, C.S. (1975). *A Vector Space Model for Automatic Indexing.* In Communications of the Association for Computing Machinery (Vol. 18, No. 11, pp. 613-620).

Santorini, B. (1990). *Part-of-Speech Tagging Guidelines for the Penn Treebank Project.* Retrieved from University of Pennsylvania Department of Computer and Information Science Technical Reports website: https://repository.upenn.edu/cgi/viewcontent.cgi?article=1603&context=cis_reports

Schler, J., Koppel, M., Argamon, S., & Pennebaker, J. (2006). *Effects of Age and Gender on Blooging.* In AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs (Vol. 6, pp. 119-205).

Seats, M. (2006). *Murakami Haruki: the simulacrum in contemporary Japanese culture*. Landham, MD: Lexington Books.

Schuler, K.K. (2005). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon* (Doctoral dissertation). ProQuest Dissertations Publishing. (UMI No. 3179808)

Stone, M. (1974). *Cross-validation choice and assessment of statistical predictions*. Journal of the Royal Statistical Society (Series B, Methodological, pp. 111-147).

Taylor, M. (Writer), & Livingston, D. (Director). (1995, July 31) The Visitor. [Television series episode] In Berman, R. (Executive Producer), *Star Trek: Deep Space 9.* New York, NY: CBS Television Distribution.

Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Katz, G., Pustejovsky, J. (2007). *SemEval-2007 Task 15: TempEval Temporal Relation Identification.* In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007, pp 75-80).

Wiebe, J.M. (1994). *Tracking Point of View in Narrative.* In Computational Linguistics (Vol. 20, pp. 233-287)

Yao, W. & Huang, R. (2018) *Temporal Event Knowledge Acquisition via Identifying Narratives.* In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18, pp. 537-547).

Zhong, Z., & Ng H.T. (2010). *It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text.* In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010, pp. 78-83).

Zhou, J., & Xu, W. (2015). *End-to-End Learning of Semantic Role Labeling Using Recurrent Neural Networks*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Vol. 1, pp. 1127-1137).

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A, Fidler, F. (2015). *Aligning books and the movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books.* In Proceedings of the 2017 Conference on Recent Advances in Natural Language Processing (RANLP 2017, pp. 803-812).

VITA

JOSHUA DANIEL EISENBERG

Born, Kendall, Florida

B.A., Mathematics, Brandeis University
B.S., Computer Engineering, Florida International University
Ph.D., Computer Science, Florida International University

PUBLICATIONS AND PRESENTATIONS

Eisenberg, J.D. "Comparison of Story Classification Methods across New Annotated Data", paper presentation at *Miami FLing 2016: Linguistics Matters,* Miami, Florida, on March 11, 2016.

Eisenberg, J.D. "Semantic Annotation using the ENVRI-Plus Ontology", presentation at *Universiteit van Amsterdam, Software and Network Engineering (SNE) Group*, Amsterdam, Netherlands, on June 2, 2016.

Eisenberg, J.D., Yarlott W.V.H., & Finlayson, M.A. (2016) Comparing Extant Story Classifers: Results & New Directions, in *Proceedings of the Seventh International Workshop on Computational Models of Narrative (CMN 2016)*, Krakow, Poland.

Eisenberg, J.D. "Comparing Extant Story Classifiers: Results & New Directions", paper presentation at *Computational Models of Narrative 2016,* Krakow, Poland, on July 12, 2016.

Eisenberg, J.D., & Finlayson, M.A. (2016) Automatic Identification of Narrative Diegesis and Point of View, in *Proceedings of the Second Workshop on Computing News Storylines (CNewsStory 2016)*, Austin, Texas.

Eisenberg, J. D. "Automatic Identification of Narrative Diegesis and Point of View", paper presentation at *Computing News Storylines 2016*, Austin, Texas, on November 5, 2016.

Eisenberg, J.D., Banisakher, D., Presa, M., Unthank, K., Finlayson, M.A., Price, R., & and Chen, S. (2017) Toward Semantic Search for the Biogeochemical Literature, in *Proceedings of the IEEE 18th International Conference on Information Reuse and Integration (IEEE IRI 2017),* San Diego, CA.

Eisenberg, J.D. "Toward Semantic Search for the Biogeochemical Literature", paper presentation at *IEEE 18th International Conference on Information Reuse and Integration, San Diego, California,* on August 6, 2017.

Eisenberg, J.D., & Finlayson, M.A. (2017) A Simpler and More Generalizable Story Detector using Verb and Character Features, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, Copenhagen, Denmark.

Eisenberg, Joshua D. "Story Detection with Simple Verb and Character Features", paper presentation at *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017),* Copenhagen, Denmark, on September 11, 2017.

Eisenberg, J.D. "Story Detection with Simple Verb and Character Features", invited talk at *Barbara Gordon Lecture & Linguistics Festival 2018,* Miami, Florida, on March 9, 2018.

Eisenberg, J.D. "Racist, Sexist, and Incoherent – The poor quality of computationally generated language", presentation at *Ignite NYC 16 – AI: Smart, Creative, Racist or Misunderstood?,* New York, New York, on April 30, 2018.