Investigations of the Properties of Narrative Schemas

A Dissertation
submitted to the Faculty of the
Graduate School of Arts and Sciences
of Georgetown University
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy
in Linguistics

By

Daniel E. Simonson, B.S.

Washington, DC
November 17, 2017

Investigations of the Properties of Narrative Schemas

Daniel E. Simonson, B.S.

Dissertation Advisor: Anthony R. Davis, Ph.D.

Abstract

Narrative schemas are generalizations of frequently re-occurring sequences of events linked through co-referring entities in text (Chambers & Jurafsky, 2009). The use of such schemas in the unsupervised analysis of text promises to assist with the characterization, comparison, and analysis of large volumes of text. However, problems exist prior to conducting such an analysis, particularly with respect to evaluation. Most work following Chambers & Jurafsky (2009) focuses on cloze task performance, which does not directly evaluate schemas. To this end, I devise techniques to directly measure properties of narrative schemas.

I first define the distinction between *score*—what is evaluated on the cloze task—and *germinator*—how a score is used to generate schemas. I re-interpret Chambers & Jurafsky (2009)'s technique for generating schemas in these terms and devise two novel schema germination techniques. These different germinators produce very different sets of schemas.

To evaluate schemas directly, I create two new tasks. The first is the *Narrative Argument Salience Through Entities Annotated* task, where schemas are shown to generally perform better than a number of baselines. I also coin a pair of *minimum description length* inspired measures. I conduct a meta-evaluation of these measures on the OntoNotes corpus (Weischedel et al., 2013); they show an insignificant degradation in schema quality despite receiving higher quality data.

I also examine the relationship between schemas, document categories, and topics. The presence of specific schemas does not predict document category well; however, document categories seem to determine the schemas generated, especially in the case of *homogenous categories*, such as the *Weddings* and *Obituaries*, which are significantly different from their *heterogeneous* counterparts. Topic models generated through latent dirichlet allocation (Blei et al., 2003) do not seem to exhibit this behavior in the model proposed here; however, topics treated similarly to document categories may provide a similar benefit.

Finally, I investigate the stability of narrative schemas, generating schemas based on 100 different ablations and cross-validations of the NYT Corpus. Generally, it seems that more documents reduce the stability of schemas, likely influenced by the endless variation of the world reflected in news narratives.

INDEX WORDS: narrative, natural language processing, computational linguistics, narrative analysis, narrative schemas

Dedication

"to the chariot of humanity,
rolling towards new horizons"

Acknowledgments

First and foremost, I have to thank the countless generations of struggle that have made it possible for me to have the opportunities and resources available to conduct this study. So many have fought namelessly for such progress; the only way to thank them, going forward, is to contribute in solidarity.

But most prominently, I must thank my parents, Karen and Gary Simonson for their unwavering support, patience, and guidance, without whom it would be impossible for me to be here today.

I must prominently highlight Tony Davis and his role as advisor in this dissertation. After taking a tutorial with him, Tony agreed to act as my advisor, completely voluntarily and without compensation. And even so, Tony's guidance, interest, and wisdom made this project largely what it is. It's difficult to thank him enough, both for the great quality of his mentorship but also for its selfless nature.

Even so, I must thank the rest of my committee as well. Both Amir Zeldes and Nate Chambers were outstanding as committee members. Amir had little specifically to do with this research topic when he joined the committee, but his boundless interest in all things linguistic made him, nevertheless, a perfect fit, and his ideas contributed greatly to strengthening this dissertation. Nate Chambers, as an external reviewer, generously supported this research as well, and given than his name appears more than anyone else's in this, he provided a significant re-assurance that what I was doing here was important and substantial while some conference reviewers missed the mark. Both Amir and Nate's feedback unambiguously made this dissertation stronger.

I must further thank the Georgetown University Department of Linguistics for its continued support of this research. In particular, Graham Katz and Paul Portner's support of me as an early student made it possible for me to be here in the first place, so I can't thank them enough.

I also owe a thank you to my partner Sylvia Sierra who has put up with me dragging this out—literally to the last minute, where I sit writing this very set of acknowledgements—but who has given and supported me with love without end. I also must explicitly thank Ryan Burke, Scott Bell, Laura Ryals, Alex Au, Joe Hsu, and the Disciples of St. Ex, who've put up with me living under a rock while I finished this, and occassionally dragged me out from underneath. Also thanks to Kristen Cornett and Edward Zaki, who both lived directly above me, for putting up with me blasting house music until 3 am some nights while I scrambled to meet deadlines.

This would also be incomplete without mentioning the many summer opportunities I was offered and accepted during my graduate studies at MITRE, Noblis, InfoChimps, and BlackBoiler. These not only made completing this dissertation possible financially, but I also learned a great deal about practical software development at each opportunity. The large scale projects I conducted during these internships contributed unambiguously to my own ability to manage projects of such scale and the lessons I learned contributed directly to managing the code base developed in this dissertation. Personally, I'd like to thank, in chronological order: Flo Reeder, Merwyn Taylor, Vince Stanford, Jean Stanford, Walt Melo, Mark White, Chris Howe, Travis Dempsey, Adam Seever, Alex Vinnik, Joe Hahn, Joe Langeway, Dan Broderick, and Jonathan Herr.

Last, but assuredly not least, I feel the need to thank some of the teachers who in past phases of my education were foundational to my development as an academic, a professional, and a person. Of course, all the educators I've had have, in one form or

another, shaped me into who I am, but a few stand-out after all these years as being particularly salient. These include, chronologically: Dawn Lawson (nee Crail), Craig Dill, Carol Hartt, Krissy Bartlett, Herr VanWassen, Susan Perryman, Sean Scully, David Bernstein, and Harold Butner.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

On the 13th of October, 2013, a woman was shot to death after a car chase from the White House to the U.S. Capitol building. Referred to as the "2013 US Capitol Shooting Incident," the title, in its bare form, entails a shooter shooting somewhere near or inside the US Capitol. Contrary to what is typical of a shooting, the instigator or suspect was not the shooter; the police were. Such subtle changes in the phrasing and framing of events can drastically alter their interpretation.

Following this, I conducted a qualitative analysis of these events, comparing four reports of the events described above from four different sources from the perspective of critical discourse analysis (Wodak & Weiss, 2003). The differences in narratives reflected a difference in underlying ideology. By approaching the problem from a narratological perspective, differences in ideologies are reflected in how storytellers characterize agents and string together events.

This qualitative analysis, however, is labor intensive. As a computational linguist, I wanted to be able to shift this analysis into the quantitative realm—to be able to look at not just four news reports, but 400,000 news reports, and to make definitive statements about ideologies and their systemic prejudices that are robustly attested in data. This does not itself eliminate the need for qualitative analysis, but it does allow qualitative analysis to scale and generalize to a degree of confidence not possible previously.

I selected Chambers & Jurafsky (2009)'s narrative schemas to attempt this sort of analysis, which appeared well suited to this task. Narrative schemas are structured generalizations of narratives derived from coreference chains and dependency parses. Their discrete nature makes their analysis more clear cut than the typical sort of probabilistic model produced as the output of many statistical NLP methods.

While it was hoped at the onset of this project that I could complete such an analysis by the end, a number of structural problems arose. The most pernicious was with respect to evaluation. How can a set of unsupervised schemas be determined to be "good" when they have never been seen before, on a data set which has no examples or corresponding schemas, or for any phenomenon like schemas for which there is no gold standard? Similarly, how can I be sure that whatever measure of "goodness" I select does not exclude extractions that are high quality and of particular interest to the analysis I seek to conduct? These two questions came to embody the core of this dissertation.

1.1 Properties as a Generalization of Evaluation

Natural language processing's evaluation paradigm has some blind spots. In this paradigm, one takes a number of examples of the phenomenon one seeks to model, feeds them into some sort of training algorithm, and derives a model that can then be evaluated on a set of holdout documents. For supervised learning, it is great; if I want to build part-of-speech tagger, such techniques are fantastic. Such a tool should need only to perform well on one and only one metric.

This is not to bash these kinds of tasks! A lot of groundbreaking work was done in this way of thinking; in fact, this dissertation would not be possible without a high quality, readily available, out-of-the-box part of speech tagger, parser, and coreference

resolution system available for use. Only now that these problems have been solved to a suitable enough degree is it possible to begin to ask the kinds of questions this dissertation sought at its outset to answer.

However, we are at a point where it begins to become possible to ask these higher order questions. In doing so, the model-single evaluation's blind spots begin to show, particularly when it comes to asking unsupervised questions. A classic example of a problem that begins to collide with this interface is that of word sense disambiguation, where prescribed definitions often fail to capture all of the senses of a word in practice, even if there is such a thing (Kilgarriff, 1997). For example, consider a system attempting to map the senses of the word "explorer" from a dictionary to text. Wiktionary[1] provides the following definitions:

1. One who explores something
2. A person who by means of travel (notably an expedition) searches out new information.
3. Any of various hand tools, with sharp points, used in dentistry.
4. (computing, graphical user interface) A visual representation of a file system etc. through which the user can navigate.

This seems a solid inventory of senses. However, how do they work out in practice? For this, I searched for "explorer" in the NYT Corpus (Sandhaus, 2008a).

> (1) "Ford *Explorer* Sport Trac ($33,330) While I find the latest version of the *Explorer* S.U.V. fairly agreeable, the pickup version seems crude and plasticky in comparison. Before Ford tries to do another new-wave truck,

[1] `https://en.wiktionary.org/wiki/explorer`

somebody in Dearborn should spend some time with the Honda Ridgeline, an oddball pickup that really works." — NYT Corpus, 1815523

(2) "THURSDAY– The singer Richie Havens and the composers Philip Glass and Galt McDermot will perform in a benefit for the Jacques Marchais Center of Tibetan Art in Lighthouse Hill, S.I. Peter Matthiessen will also read from his work at the event at the *Explorer*'s Club, 46 East 70th Street, from 7:30 to 10 P.M. Tibetan hors d'oeuvres and a buffet dinner will be served. Tickets, $150, from (718) 987-3500. Conservators' Gala" — NYT Corpus, 636965

(3) "9 A.M. (NBC) MACY'S THANKSGIVING DAY PARADE – They call it the longest running show on Broadway. This year the parade celebrates its 80th birthday with an expanded selection of floats and balloons (including Dora the *Explorer*, above), and the premiere of the Macy's Great American Marching Band, made up of students from the 50 states and the District of Columbia. Guests of honor include Helen Gross, now 101 years old, who in 1927 and 1928 was the first – and only – queen of the parade." — NYT Corpus, 1806768

(4) ""National Geographic *Explorer*" turns its attentions to a human species as it visits Buenos Aires to catch some tango aficionados in the throes. The scenes of professionals, notably Miguel Anjel Zotto and Melena Plebs, waging what is called "a battle of legs," are easy to look at. Harder to take is the puffy narration (the tango is described as being "about love and longing and loss")." — NYT Corpus, 633711

Examples (1) and (2) both use the term "explorer" in naming something, but the name of that thing itself only invokes the notion of exploration and explorers

for sake of association, and is not necessarily the thing itself or a condition for its exclusive use. (1) is the name of an automobile that is intended to make the driver feel like an "explorer;" (2) is the name of an upscale social venue that serves exotic hors d'oeuvres. (3) refers to a balloon, albeit a balloon of a cartoon character who educates children through exploration-themed narratives. (4) refers to a television show that catalogs travel. While each of these invoke some notion of the original sense of "explorer," the things to which they directly refer only shares partial relationships with the conventional meaning of the word.

These examples were not terribly difficult to find, and they show what I was attempting to demonstrate: that even with a whole dictionary worth of word senses, human speakers will continuously and indefinitely utter words in senses that are unexpected and novel, not to mention coin entirely new words on the fly. Especially within the realm of semantic meaning and the discursive processes that depends on such entailments, the target function becomes increasingly difficult to catalog and illustrate through discrete, pre-meditated classifications and examples alone. That said, the problem of word sense disambiguation is best left to unsupervised models, as with understanding the narrative structure of text.

However, when stepping into unsupervised circumstances, it becomes extremely difficult to devise and conduct a single, exhaustive evaluation on the products of a model. Often the trick to doing an unsupervised learning task is figuring out a clever way to evaluate what was learned, such as Chambers & Jurafsky (2008, 2009)'s use of the cloze task or Yarowsky (1995)'s "pseudowords" for word sense disambiguation. Unfortunately, these sort of solutions limit the set of problems we can solve to those we can come up with such evaluations for. It also leads to a singular focus in improving performance on such tasks, such as the literature now dedicated to models which gain

incremental improvements on the cloze task (see Section 2.8), showing ever decreasing interest in the narrative schemas originally proposed by Chambers & Jurafsky (2009).

Again, this is not necessarily a bad thing. Such models provide hints in their parameters and innovations about the structure of discourse, albeit often in architectural changes that are intertwined. However, such singular focus is often the opposite of what a qualitative analysis seeks to achieve. A lot of qualitative analysis begins by trying to understand the properties of previously unseen data and not knowing what to expect. When new findings somehow explain some previously unexplained aspect of the world, this new insight is powerful and valuable. This is not done by grinding up performance on some convenient measure.

Thus, this puts us at an impasse. Leveraging quantitative tools to this end could drastically accelerate such processes. At the same time, these new insights need to be of high quality—robust and reliable. For discoveries to be rigorous, we cannot accept all novelties as necessarily good. The best way to do this—the approach eventually taken by this dissertation—is to instead conduct a variety of evaluations to understand the properties of the phenomenon being understood. Unlike traditional evaluation, none of these are specific comparisons between the thing generated and examples of what it should be. Rather, they should define expectations about the reliability and properties of established tools and techniques being deployed into unknown conditions.

In other words, we want to not necessarily evaluate our analytical tools in the traditional sense, but rather, we want to understand the broader set of properties of those analytical tools.

One way to think of this is as a generalization of the idea of evaluation. For a supervised algorithm, you want to understand one and only one property of a given algorithm. For a POS tagger, for example, you only really care about its accuracy. It

does not really matter that a POS tagger does not identify salient entities well in a text or its behavior degrades when given fewer training documents. In creating POS tagger, the score needs to be as high as possible, and all other properties are really irrelevant. This is not a bad thing; it is just that for this specific problem, this is the sole property that matters.

But for conducting a novel analysis using unsupervised learning, many different properties must be assessed because the one you really want to know, you cannot know, because by definition it is unknown—that is why the problem is unsupervised in the first place.

In the context of this dissertation, beyond the definition of where a narrative schema is derived from, the assumptions made to come to that definition, and a few examples given by Chambers & Jurafsky (2009) and Balasubramanian et al. (2013), there are few things really known about the properties of narrative schemas. What to expect from that definition is unknown, and that is a good thing. The schemas themselves can tell us new things about the language data they have been learned from—and they do! However, a traditional supervised learning paradigm would exclude some of the more interesting results *because they are new and interesting*. Thus, a more property centric paradigm can provide us with some ideas about how robust a set of schemas might be. Perhaps more importantly and more directly, focusing on properties allows us to select those that are more important for a specific analysis, to improve performance with respect to those properties, and to use the values of those properties to bound the certainty of a novel analytical result.

Therefore, in pursuit of improving schemas for unsupervised analysis of text and improving their evaluation in light of that goal, this dissertation seeks to define as many of these properties as possible to better understand narrative schemas and how they might best be employed in an unsupervised analysis of new text.

1.2 Dissertation Outline

To this end, I conducted a number of experiments on narrative schemas. These are divided into four analysis chapters (Chapters 3 – 6), each with specific goals in mind.

In Chapter (2), I summarize the existing literature on script knowledge, narrative schemas, computational implementations of such techniques, and the evaluations used.

In Chapter (3), I define a distinction between *score* and *germinator* in schema generation. I redefine Chambers & Jurafsky (2009)'s technique for generating schemas in these terms—referring to it as *linear induction*—and I devise two new schema germinators: *counter-training* and the *random walker*. I show examples of the schemas that result from these processes, and perform a qualitative analysis of the outputs.

In Chapter (4), I define two new evaluations for schemas that evaluate them directly. The first is the *narrative argument salience through entities annotated* or *NASTEA* task, where schemas are used to select salient entities as annotated in the New York Times corpus (Sandhaus, 2008a). The second is a pair of information-theoretically inspired *minimum description length* or *MDL* measures. Both of these rely on a common *presence* measure that measures how precisely a schema has been instantiated within a document. NASTEA shows some promise, differentiating schemas generated with the germinators developed beating a series of simple baselines, albeit by at best a few percentage points. MDL is meta-evaluated on the OntoNotes corpus and appears to perform insignificantly worse on gold standard data; its event centricity suggests that it is vulnerable in many respects to the same flaws as the cloze task.

In Chapter (5), I analyze the relationship between narrative schemas and document categories. Overall, it seems that document categories predict what sort of

schemas you get, but the relationship does not seem to work well in reverse. I also test whether topics can act as substitutions for hand-curated document categories in the context of schema generation. The findings here do not support such substitutions in the model developed here with topic models in mind. Further investigation suggests treating topics like document categories, which may provide similar benefits to document category siloing.

In Chapter (6), I assess the stability of narrative schemas by systematic changes to the training data used to generate schemas. In the process, I generated 3,978,865 schemas using 100 different samples of the training data across three different algorithms. It seems that schemas become more unstable when more documents are added to the training data, suggesting that the problem of schema generation becomes more difficult as it must disentangle more narratives.

In Chapter (7), I conclude the dissertation, reflect on the findings, and re-address the issues raised in this chapter.

1.3 Schemas in Practice

If you, the reader, wish to deploy narrative schemas in an application, I recommend the following.

First, it is important to ask yourself whether narrative schemas are actually relevant to the problem you are working on. Narrative schemas are sets of events—as expressed through verbs in this implementation—with shared participants in overt syntactic slots directly connected to event verbs. A full technical description is in Section (2.7.1). Figure (1.1) contains an example of one of these generated using the techniques described in this dissertation. The red square indicates someone who was

arrested, charged, and indicted with something by the blue circle. Additionally, the red square testified for the blue circle and cooperated with them.

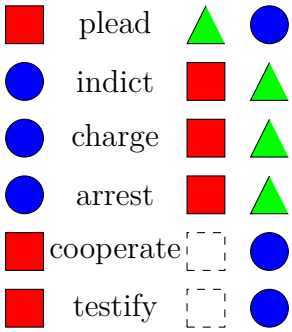


Figure 1.1: A narrative schema generated using the techniques described in this dissertation. Shared event slots share a common shape and color. Each column represents a type of event slot: the first column is the subject slot, the second column is the event verb, the third column is the object slot, and the fourth column is the general preposition slot.

The following ought to be true if you want to use schemas in your analysis:

1. Does your problem involve language data?

2. Are there parsers and coreferencers available for the natural language that your data is expressed in?

3. Does your language data involve similar sequences of events repeated in different documents?

4. Do these sequences of events share participants, particularly in cases where those participants are overtly mentioned in syntatically expressed subject and direct object slots?

5. Is the collocation of events meaningful for the results you are trying to obtain?

If you answered yes to all of these, narrative schemas may prove helpful.

If your data has document category annotations, I recommend separating them by document category before generating schemas. If your data does not have document category annotations, preliminary results indicate to build a topic model and silo the data by topic.

You will need to consider the categorical context of your data. If your data is repetitive—to the extent that it is written from a template—then it is homogeneous. Depending on how long your schemas are, they will encapsulate documents based on few—possibly one single—schema. If your data more resembles typical news, however, containing narratives of varying structure and content, your data will be heterogeneous. This means you will need multiple schemas per document to cover the contents.

Choosing a germinator can be tricky business, and many parts of this dissertation are dedicated to measuring properties of schemas generated by different germinators. If speed is your primary concern, I recommend Chambers & Jurafsky (2009)'s germination method, here referred to as linear induction. Without careful parameter tuning, though, this technique can result in a long tail of schemas containing a single event, and associations are not globally maximized and are sensitive to the order in which events are considered. If you want to obtain schemas that have a strong distributional resemblance to the original data, I recommend the random walker. Many schemas generated by the random walker are repetitive, however, because random selections have a tendency to replicate the Zipfian distribution of natural language text. If stability and consistency are your primary concern, counter-training proved most stable and scored higher than its stochastic counterpart in the most rigorous tests executed here on the NASTEA task. Counter-training is very slow, though, as all decisions to add events are considered simultaneously. Both the random walker and counter-training have more parameters than linear induction as well. That said, all germinators deliver different generalizations of the data, and there is no real reason

not to pool all of their outputs together for an unsupervised analysis. When you do not know the right answer to the problem you are trying to solve, there is no reason—except perhaps expediency—to not consider all possible angles to a solution.

1.4 Tools Used

A number of different NLP and machine learning tools and packages were leveraged at various times during this dissertation. In this section, I provide a brief description of the various tools used. Parts of this set of tools will be referenced as they are implemented uniquely in different experiments throughout the dissertation.

The vast majority of the code for the experiments described in this dissertation is written in Version 2.7 of the Python scripting language (Rossum, 1995). This code is written predominately in a functional style, made possible by the powerful primitives provided by Python.

For NLP preprocessing, I used the Stanford CoreNLP suite of tools (Manning et al., 2014)[2] The same version of CoreNLP was retained through the dissertation to maintain a consistent foundation—changing the preprocessing component could make it difficult to isolate the source of iterative improvements and changes through the dissertation. Primarily, CoreNLP was chosen for having both parsing and coreference facilities (de Marneffe et al., 2006; Lee et al., 2013) trained and readily useful. The full pipeline includes pre-requisites for those tasks, including but not limited to tokenization and part of speech tagging (Toutanova et al., 2003; Toutanova & Manning, 2000), both of which are refered back to in performing higher-level information extraction tasks. The named entity annotations provided by the CoreNLP Pipeline are leveraged throughout this dissertation as well (Finkel et al., 2005). The specific applications of these tools will be discussed as relevant.

[2]Stanford CoreNLP, Version 3.4.1 (2014-08-27)

In some circumstances, a small quantity of information must be tokenized and normalized—for example, the NYT metadata. For this, rather than deploy the whole CoreNLP pipeline to extract a few tokens, the easy-to-deploy NLTK library is used in these circumstances (Bird et al., 2009).[3]

For LDA topic models (Blei et al., 2003), the implementation deployed in gensim is used (Řehůřek & Sojka, 2010).[4] Gensim is efficient and easy to deploy in Python.

1.5 Open Source Software and Materials

An open source version of the software developed for this dissertation is available at:

`https://github.com/thedansimonson/durruti`

For licensing information, please refer to the readme and license files in the repository.

Additionally, the narrative schemas generated in experiments in this dissertation are available at:

`http://schemas.thedansimonson.com/`

[3]NLTK, Version 3.2.1

[4]Gensim, Version 0.13.3

Chapter 2

Literature Review

2.1 Introduction

In this dissertation literature review, I explore the prior work in script-based and schema-based models within natural language processing, starting with the theoretical underpinnings of such work in both narratology and cognitive linguistics, exploring work from the late 1970's to the present state of the art in such models.

Much of this work is defined by the tools available to the researchers who built them. Statistical models developed over the past 30 years have dramatically changed the nature of natural language processing and the capabilities of researchers working therein. Specifically, named entity recognition, parsers, coreference, generative modeling and others have improved dramatically; as this literature review will show, these improvements have changed the nature of the work on the problem of script inference. While related, much of this work has aimed for different goals as well—creating models that optimize a specific metric or task, or to produce some sort of end product, such as schemas or event chains. Evaluations have varied between work as well, though this is a generally closed set.

I begin in Section (2.2), I discuss some narrative theory that will be used to describe aspects and phenomena within the scope of this dissertation. In Section (2.3), I discuss the pre-statistical work on schemas and story understanding, including the theoretical underpinnings of work fundamental to this dissertation. In Section

(2.4), I discuss the problem of story understanding. In Section (2.5), I discuss some related supervised challenges, including MUC. In Section (2.6), I discuss evaluations of narrative models devised and used throughout the latest boom in literature on modeling script knowledge. In Section (2.7), I discuss work that generates actual narrative schemas in one form another. In Section (2.8), I discuss work that seeks solely to perform well on the cloze task. In Section (2.9), I discuss overall trends within the existing literature.

2.2 Some Narrative Theory

Bal (1997) provides an introduction to narrative theory. Some the vocabulary will be essential in describing what narrative schemas or commputational models of narrative actually contain, from a linguistic perspective. It has been cited in other computational work on narrative (B. Miller et al., 2015; Vossen et al., 2015) and provides a bridge to that work through a common theoretical framework.

Bal's model consists of three separate layers: the *text*, the *story*, and the *fabula*.

The most abstract layer is the *fabula*: this is the logical description of events entailed in a particular story. It includes the agents and the interactions between agents through a sequence of events distributed through time in a particular order with particular durations.

An author must then make a series of choices—*aspects*—that adapt a fabula from a bare sequence of events to a *story*. These include:

- ordering the sequence of events, potentially in an order that does not match the chronological order defined in the fabula

- the quantity of information[1] given to each element of the fabula
- turning actors into characters through the "distinct traits" attributed to them by the narrator
- also giving locations traits, them becoming "specific places"
- other relationships not indicated in the fabula are added, including: "symbolic, allusive, traditional, etc."
- selecting a "point of view" to narrate the story from

While the fabula describes the events and actors themselves, the aspects are choices that are made in preparing them for presentation, making them a story rather than a dry enumeration. Even so, such an enumeration constitutes a certain point of view, like the "objective" point of view that news typically attempts to take.

The *text* is the actually realized transmission of a story—either through prose, film, paint, porcelain, etc.

Bal describes these in detail in reverse order, starting with the text. The reason for this is that narrative is predominantly a receptive process–whatever the author's intention may have been, an analysis or reading of a narrative begins with the text it is embodied in, which then is eventually rendered as a fabula in the mind of the analyst or reader.

Coincidentally, this is beneficial for applying Bal's analysis to the quantitative analysis of narrative, as quantitative approaches too must begin with the text itself. In some sense, computational work on narrative can be thought of as extraction of fabulae from text, and much information extraction work, such as event extraction, can be thought of as aspects of fabula extraction.

[1]Bal refers to this as "the amount of time which is allotted," but in static media, such as prose or a painting, the quantity of information definition more broadly covers these media.

2.2.1 Understanding Schemas within Bal's Framework

It is debatable what exactly a schema is within Bal's framework, and it depends largely on how rigidly "fabula" and "schema" are defined. Since a fabula contains specific characters and events, schemas should not be confused with fabulae, as a schema contains a generic representation of similar events, and while it may refer to specific types of actors, it doesn't refer to any specific individuals, which is what a fabula contains.

If one is more loose with the degree of specificity required, then a schema can be thought of as a re-occurring fabula—a sequence of events occurring over and over again, in each case the specific individuals filling the roles of actors and locations being only a storying of that fabula. In other words, a schema can be thought of as a *type* while a fabula is a *token* of a specific schema. This seems to be a weaker interpretation of Bal, albeit one that's tempting, as it keeps the theoretical machinery required to describe narrative small. However, given that many narratives are long and feature twists and turns of novelty, it may be the case that narratives themselves are composed of many schemas, patched together to form the story of being told.

In either case, Bal's receptive focus from text to fabula—over say, a productive one—fits nicely with a statistical, computational framework, wherein reception of the text is the source of the model or analysis. This sort of macro-reading—as used by Mitchell et al. (2009)—is different from a typical human, "micro-reading," but the direction implied in the theory holds and remains relevant.

This theoretical debate I would like to leave open for later work. For the purposes of limiting the scope of this dissertation, narrative schemas will be considered clusters of similar fabulae. Fabulae will be strictly the specific logical events and actors that occur in a specific text.

2.3 Frames, Scripts, and Pre-statistical Story Understanding

While Bal (1997) provides a narratological perspective on the human understanding and linking of events, cognitive linguistics provides another analysis of the problem with broader goals—to understand the broader human interpretation of events, both within narratives and through the day-to-day course of our lives.

Work to this end spurred Schank and Abelson (1977) to devise a theory of scripts as an explanation for human understanding of sequences of events. This work included hand curation of such scripts. This spurred a number of research directions, some of which are described below.

2.3.1 Schank and Abelson (1977)

The origin of this enterprise is in the work of Schank and Abelson (1977), who proposed a system for conceptualizing and enumerating what we know when we understand something. In their framework, we must possess knowledge of *scripts, plans, and goals*. *Scripts* are generalizations of recurring episodes. *Plans* are how two events are connected together when no available script can join them. *Goals* are the driving forces for such plans. Schank and Abelson (1977) argue that these components are essential in developing a computer system capable of understanding. This is, of course, an extremely difficult problem. Given the complexity and variety of human knowledge, such a research program could extend without limit. Therefore they limit their problem scope to "the world of psychological and physical events occupying the mental life of ordinary individuals, which can be understood and expressed in ordinary language... the common sense (though perhaps wrong) [sic] assumptions which people make about the motives and behavior of themselves and others..." This common

sense, it is hoped, can be programmed to provide intuition to a non-human observer or participant in the human world. They go on to state that

> "...more is at issue than 'semantics.' It is 'pragmatics,' the way things usually work—not how they might conceivably work—which most often impels the reader toward an interpretation. The reader brings a large repertoire of knowledge structures to the understanding task."

In other words, Schank and Abelson (1977) seek to create a representation of this knowledge that is typical or implicit in a situation or given description of a situation.

Scripts represent *specific knowledge* about frequently experienced events, a sort of episodic memory: "When a standard repeated sequence is recognized, it is helpful in 'filling in the blanks' in understanding. Furthermore, much of the language generation behavior of people can be explained in this stereotyped way." This idea of episodic memory is contrasted with "the more scholastic notion of semantic memory," an ontological one, which they claim "the complexity of the possible combination of elemental concepts makes this extremely cumbersome." In other words, the mental representation of memory as a generalization of episodes is necessary; without such, the memorization of events would be too computationally cumbersome to perform without some sort of generalization mechanism. While this notion was devised in the context of computers with exponentially smaller storage capabilities, the volumes of language data computers are expected to handle have grown comparably with the size of storage media, and even with the fastest storage media available, such massive volumes of data require some sort of distillation to perform tasks quickly.

Schank and Abelson (1977) posit two special mechanisms based on the fact that people are able to recognize such sequences of events. First, people can identify a script from a handful of salient events with respect to that script; second, "a script

applier" that uses the script to fill in voids in the causal chain implied by the original salient sequence of events. These two mechanisms allow for pragmatic inference across parties, since through the scripts, both parties can infer what the other knows.

My choice in using a schema-based analysis of text is in part motivated by Schank and Abelson (1977)'s own intuition about the applicability of their work to understanding and interpreting discourse, that "Lurking beneath the surface, however, is an interest in the ingredients of personal belief systems about the world, which dispose people toward alternative social, religious, or political actions." This undertone of Schank and Abelson (1977)'s model of knowledge and cognition largely informs my own adoption of Chambers and Jurafsky (2009)'s methods, which will be discussed later.

The Restaurant Script

Perhaps the best known Schankian script is the "restaurant script," described in detail in Chapter 3 of Schank and Abelson (1977). It is a thorough—though by their own admission, incomplete—mapping of the episodic knowledge of visiting a restaurant.

Events are denoted in Schank (1973)'s *conceptual dependency* (CD) theory, a model for representing what sentences contain about events through a constrained set of abstract primitives, similar to other semantic decomposition approaches such as natural semantic metalanguage (NSM) theory (Goddard & Wierzbicka, 1994). Events in conceptual dependency theory are captured by decomposition to specific *primitive acts*. Those relevant to the restaurant script include `PTRANS`, `ATRANS`, `MOVE`, `INGEST`, `MBUILD`, `DO`, `ATTEND`:

Figure (2.1) contains Schank and Abelson (1977)'s own original rendition of the restaurant script. The script itself contains aspects of a conventionalized restaurant encounter, broken down into atomic, conceptual dependency theory primitive acts.

Table 2.1: CD primitive acts contained in the restaurant script. Table from (Rich & Knight, 1991, 278).

ATRANS	Transfer of an abstract relationship (e.g. give)
PTRANS	Transfer of the physical location of an object (e.g. go)
MTRANS	Transfer of mental information
MOVE	Movement of a body part by its owner (e.g. kick)
INGEST	Injection of an object by an animal (e.g. eat)
MBUILD	Build new information out of old (e.g. decide)
ATTEND	Focus a sense organ toward a stimulus (e.g. look)

Consider, for example, in Scene 2, the subsequence:

$$\begin{aligned}
&1)\text{S } \textbf{MTRANS} \text{ food list } \textbf{to } \textbf{CP}(\text{S}) \\
&2)\text{S } \textbf{MBUILD} \text{ choice of F} \\
&3)\text{S } \textbf{MTRANS} \text{ signal } \textbf{to } \text{W} \\
&4)\text{W } \textbf{PTRANS} \text{ W } \textbf{to } \text{table} \\
&5)\text{S } \textbf{MTRANS} \text{ 'I want F' } \textbf{to } \text{W}
\end{aligned} \tag{2.1}$$

This sequence (2.1) indicates the customer, S, commits the food list from the menu to memory (1), uses that new information to develop a choice from that menu (2), hails the waiter with the mental transfer of a "signal" (3), after which the waiter physically moves themself to the table (4), then the customer conveys the choice made in (2) to the waiter (5).

While thorough, there are flaws in this script's design. For example, Schank and Abelson choose to portray the selection of a food item as "choice of F," with the instantiation of an item F—the class of food desired. However, there's no actual food in the scene yet—S has created a desire for F, not F itself. Cumbersomely, F is

hedged in hard-coded modals throughout the sequence: "choice of F" and "'I want F'," a specific choice of wording that is expressed in a number of ways. Rather than representing *the desire* as an object within the script, it is written around. This trickles down the sequence, with this desire handled differently by the next short sequence as "W **MTRANS** (**ATRANS** F) **to** C," where the waiter expresses the desire for F to the cook C. This is nitpicking, but it shows the difficulty of devising a script by hand in a way that is consistent from all linguistic perspectives.

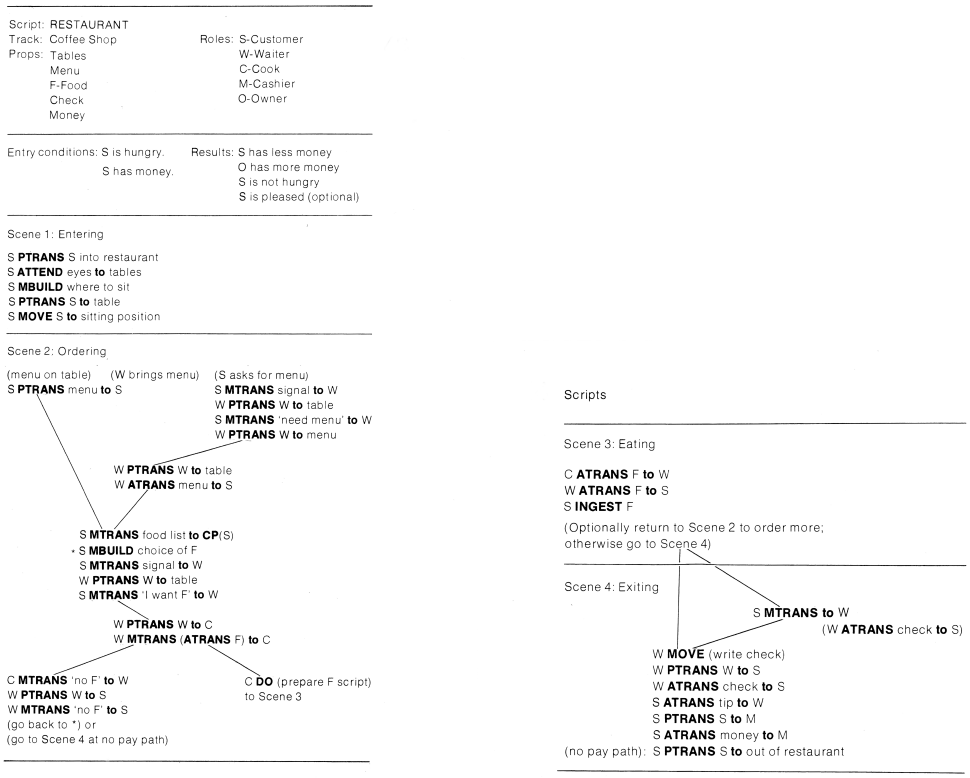


Figure 2.1: The restaurant script, as illustrated in Schank and Abelson (1977).

It's also worth noting, this is a typical sequence of events, but not necessarily followed precisely. The script itself contains optional events, so it is actually a lattice

rather than strictly being a set of chains. Other intervening events may occur—for example, conversations and other interruptions. Rather, these do not invalidate the script itself and pragmatically, such exceptions to the script are made worth being mentioned through such a violation. While this dissertation does not delve directly into the interaction of pragmatics and schemas, it is worth considering in any evaluation framework, especially juxtaposed against the solutions presented in this dissertation (c.f. Chapter 4).

2.3.2 Obtaining Frames

Schank and Abelson (1977) posit the existence of scripts and that a discrete, finite set of them exist, through which events in the world are interpreted. For these to be used in a practical sense—that is, in a computational or quantitative analysis—some set of frames, scripts, or schemas must be acquired for such a system to function. Work following Schank and Abelson (1977) did this in two ways: unsupervised learning and hand-curation.

In motivating the unsupervised approach to extracting schemas, Mooney and DeJong (1985) point out, "to process a wide range of text, a schema-based natural language processor must possess many schema[s], perhaps hundreds of thousands," and furthermore, that these schemas must be obtained through an automated process, since "there are simply too many" and because "hand coding does not allow for dynamic augmentation of world knowledge." That is, if Mooney and DeJong (1985) had chosen to build a set of schemas by hand, their 1980's schemas would fail to understand modern phenomena, such as "internet trolls," "hacktivism," and "smart phone addiction"—the product of the Herculean task of building those hundreds of thousands of schemas would be obsolete before the task could even be completed. The

hope, then, is that an automated system can learn these just as people do—through "reading" news.

Given the potential utility of schemas, many works pursued this end (DeJong, 1983; Mooney & DeJong, 1985; Norvig, 1983; Wilensky, 1983). Mooney and DeJong (1985) provide a good example of how these systems worked. Typically, a parser of one form or another would process documents, then the parses are used devise a causal chain for each document. These causal chains are generalized into schemas if narratives containing similar events, with actors oriented toward similar goals, are not yet accounted for with an existing schema. The process of combining these similar chains produces a "general causal structure which achieves a common goal."

Of course, while Mooney and DeJong (1985) suggested that the task of building a library of frames was too difficult for mortals to complete, it did not stop others from trying (e.g. Minsky (1974)). The most prominent example being FrameNet (Baker et al., 1998; Fillmore et al., 2003), which attempts to capture word meaning through *semantic frames*:

> "schematic representations of the conceptual structures and patterns of beliefs, practices, institutions, images, etc. that provide a foundation for meaningful interaction in a given speech community" — (Fillmore et al., 2003)

These frames differ from schemas or scripts in noteworthy ways. While schemas and scripts detail an abstract sequence of re-occurring events, a frame details knowledge of a single type of event and its participants. For all intents and purposes, a schema in effect contains multiple frames. This, in theory, reduces the complexity and permutations of annotations required to complete annotation of the full space of frames. A schema or script typically connects what would be defined as multiple frames

together. In FrameNet, frames surround what might constitute a single event in a schema in greater detail, and the roles contained in that frame are more precisely defined. While no `restaurant` or `service encounter` frames have been specified in the current version of FrameNet, `Renting`[2] offers an example comparable to a service encounter.

Documenting these frames is not done completely manually, as some automated procedures are used, but the final product is hand-curated. The end product is a representation of these frames, derived from the syntactic and semantic properties of the words that compose them. Work on FrameNet continues through the present (Ruppenhofer et al., 2006), as is necessary for such a system to avoid obsolescence. Some external efforts have been made toward automatic extraction of frames as well. For example, Green and Dorr (2004) induce frames from WordNet senses using a measure of the density of senses in a given subtree of the ontology; in a manual evaluation of the induced frames, humans generally considered 88% of the induced frames "to convey information about the same, similar, or a closely related situation."

2.4 Story Understanding

Story understanding, at its heart, is about understanding the events entailed in a specific story and representing the knowledge contained therein. Work to such an end has been around since the pre-statistical days of natural language processing, reflecting trends within the field but also the narratological work of the time as well. More recent work reflects the rise of the digital humanities, with natural language processing techniques enabling new types of studies of corpora of narrative texts.

[2] `https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Renting`

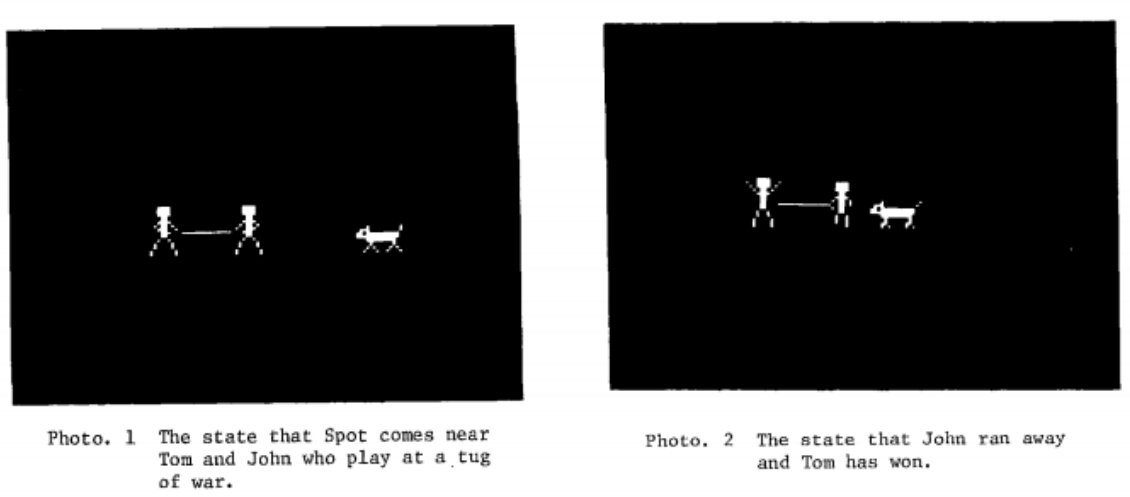


Photo. 1 The state that Spot comes near Tom and John who play at a tug of war.

Photo. 2 The state that John ran away and Tom has won.

Figure 2.2: Sample of the simulation used in Ogawa et al. (1980) to study story understanding.

Ogawa (1980) provides a classic, pre-statistical example of a story understanding paper. Here, a hand-curated system composed of "micro-actors" attempts to communicate to one another, considering story understanding a problem of "knowledge representation, and the mechanism of understanding and inference." Micro-actors are essentially problem solving modules that communicate with one another, each composed of a set of instructions for processing a message received and a set of statements of declarative knowledge. If Ogawa's system is given a large problem, it divides such a problem into smaller subtasks, which are distributed amongst the micro-actors, and which communicate solutions to one another. Ogawa (1980) thus treats story understanding as a problem of micro-actors embodying the entities contained in the story and traversing the states they are described to adopt. Ogawa adds "demon functions"—tests of certain situations—to the micro-actor framework to assist in this

regard; the system is tied together through a "Sinchronizer [sic]" micro-actor that simulates the time and space that the micro-actors co-exist in.

In the mid-1980's, computing power and memory had increased enough to make it viable to deliver statistical solutions to NLP problems. Focus shifted to problems that were more linguistically fundamental and susceptible to approaches that involved counting and aggregating knowledge from lots of small examples: POS tagging, parsing, document classification, probabilistic models, and lexical semantics (Hall et al., 2008). With this, story understanding, having been a topic based on thoroughly hand-curated models of narrative, fell into decline, as seen in Figure (2.3).

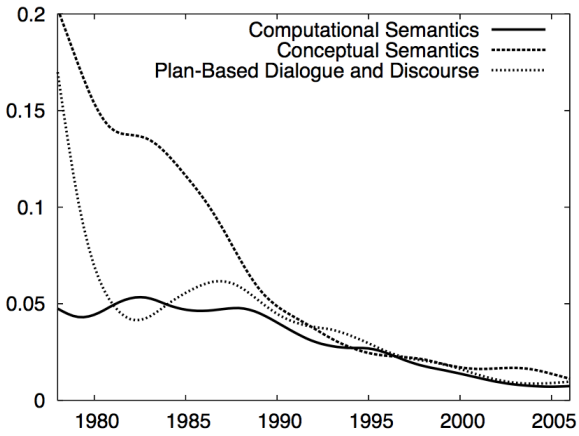


Figure 2.3: Figure from Hall et al. (2008) showing the decline of story understanding under the topic heading "conceptual semantics." They used modern topic modeling techniques to conduct a meta-analysis of older papers in NLP.

However, with the rise of the "digital humanities," a number of symbolic frameworks for representing narrative structures have been developed, which feature some form of human annotation to augment or develop the system's capabilities (Caselli & Vossen, 2016; Elson & McKeown, 2009, 2007; Vossen et al., 2015). These frameworks contain formal standards for encoding events, actors, causality, time, and other information, implicit or explicit, interpreted and understood by human readers. Vossen et

al. (2015) and Caselli et al. (2016) develop an extraction and annotation framework centered around the climax of a narrative, with other events specified as bridging relations to the climax. Elson & McKeown (2009, 2007)'s SCHEHERAZADE system encodes "narrative structure rather than the mechanics of the story-world" to circumvent the world knowledge and complexities required to do so.

Additionally, much of the digital humanities literature presents a number of techniques that do not use schemas or script-style structures to analyze narratives. These often leverage information from a number of different NLP techniques, but often in ways that are *ad hoc* for the problem at hand. Reiter et al. (2014) use a number of techniques to determine similarity between a number of hand-written transcriptions of oral narratives, performing document alignment and clustering using a whole suite of NLP tools and knowledge bases, such as WordNet and FrameNet. Elson et al. (2010) extract social networks from within literary fiction, finding that the amount of face-to-face interactions does not "diminish as the number of characters in the novel grows," overturning contrary claims of literary scholars in more shallow studies. Miller et al. (2015) also perform an alignment procedure but on 24 non-fiction texts relating to a single event. They found that many of the techniques designed for fiction are poor at handling the large number of entities contained in news text—much greater than that of fiction.

2.5 SUPERVISED CHALLENGES

While story understanding declined in the 1990's, a set of *template filling* tasks in part filled the void, particularly the *Message Understanding Conference* (MUC) (Grishman & Sundheim, 1996) part of the TIPSTER program (NIST, 2014)—used a set of pre-

annotated templates to conduct information extraction with the goal of filling templates that contained specific roles contained in similar types of stories.

MUC originated through DARPA funding to promote research on automatic analysis of military messages (Grishman & Sundheim, 1996), particularly through the automatic filling of a set of templates using hand-curated training data. Figure (**??**) shows one of these templates. According to Grishman and Sundheim (1996), MUC-1 was largely exploratory, with the template-filling aspect of MUC crystalized in MUC-2. Both of these involved reports of naval engagements. MUC-3 and MUC-4 shifted the focus to reports of terrorist events in Latin America. MUC-5 added nested templates, greatly increasing the complexity of the task. MUC-6 strove to drive the technology into a more task-independent direction. MUC-7 added named entity evaluation across multiple languages and was the final iteration of MUC (Chinchor, 2001). Notably, in each instance of MUC, the type of documents—and consequentially, the template that needed to be filled—was constrained. The task of template identification—figuring out which template needed to be filled out of a set of templates—was not necessary in this context. By MUC-7, while the manual template annotation task had been refined to obtain reliable results with F-measures greater than 90%, automatic extraction F-measure scores remained at best around 40% (Marsh & Perzanowski, 1998).

Starting in 2005, the Recognizing Textual Entailment (RTE) challenge provides another sort of challenge based on a specific data set—in this case, recognizing text that entails other text (Sammons et al., 2012). For example, if considering the following:

> "The purchase of Houston-based LexCorp by BMI for $2Bn prompted widespread sell-offs by traders as they sought to minimize exposure." — (Sammons et al., 2012, 3)

```
McCann has initiated a new so-called
global collaborative system, composed
of world-wide account directors paired
with creative partners. In addition, Pe-
ter Kim was hired from WPP Grout's J.
Walter Thompson last September as vice
chairman, chief strategy officer, world-
wide.

<SUCCESSION_EVENT-9402240133-3> :=
    SUCCESSION_ORG : <ORGANIZATION-9402240133-1>
    POST: "vice chairman, chief strategy
    officer, world-wide"
    IN_AND_OUT : < IN_AND_OUT-9402240 i33~5>
    VACANCY_REASON : OTH_UNK
< IN_AND_OUT-9402240133-5> :=
    IO_PERSON : <PERSON-9402240133-5>
    NEW_STATUS : IN
    ON_THE_JOB : YES
    OTHER_ORG : <ORGANIZATION-9402240133-8>
    REL_OTHER ORG : OUTSIDE_ORG
<ORGANIZATION-9402240133- i> :=
    ORG_NAME : "McCann"
    ORG_TYPE : COMPANY
<ORGANIZATION-9402240133-8> :=
    ORG_NAME: "J. Walter Thompson"
    ORG_TYPE : COMPANY
<PERSON-9402240133-5> :=
    PER NAME: "Peter Kim"
```

Figure 2.4: An example of a MUC template (bottom) and the text it was generated from (top) (Grishman & Sundheim, 1996).

a system should be able to recognize that "BMI acquired an American company" is entailed by it, but not "I like turtles" or the more difficult "BMI acquired a French company." The task originated as the PASCAL RTE challenge, and the data set is often referred to as such. The Text Analysis Conference (TAC) adopted it as a track in 2008 (Bentivogli & Giampiccolo, 2011).

Critiques have been made of the PASCAL data set, specifically that many of the textual inferences defined are not actually entailments or even implicatures of the given text, but rather are much more sophisticated relationships between bits of text which were rendered by how the data set was produced (Zaenen et al., 2005). The Stanford Entailment Corpus (Bowman et al., 2015) is largely regarded as the next generation of the PASCAL dataset.

2.6 Evaluations of Script Models

Thus far, work on narrative schemas has focused on optimizing the performance on the narrative cloze task. I describe this in Section (2.6.1), as well as the recently developed story cloze task (Section 2.6.2) and their respective limitations.

2.6.1 The Cloze Task

Chambers and Jurafsky (2008; 2009) evaluate their models with the *narrative cloze task*. This entails removing a verb from a sequence of verbs that are tied together through a coreference chain—a sequence of utterances that refer to the same entity, often nouns but potentially other sorts of mentions—and seeing how well the narrative model can guess as to the verb contained in the missing verb slot.

Specifically, the procedure works as follows: take a coreference chain from a document, remove one event verb from the chain, score all verbs in the corpus according to

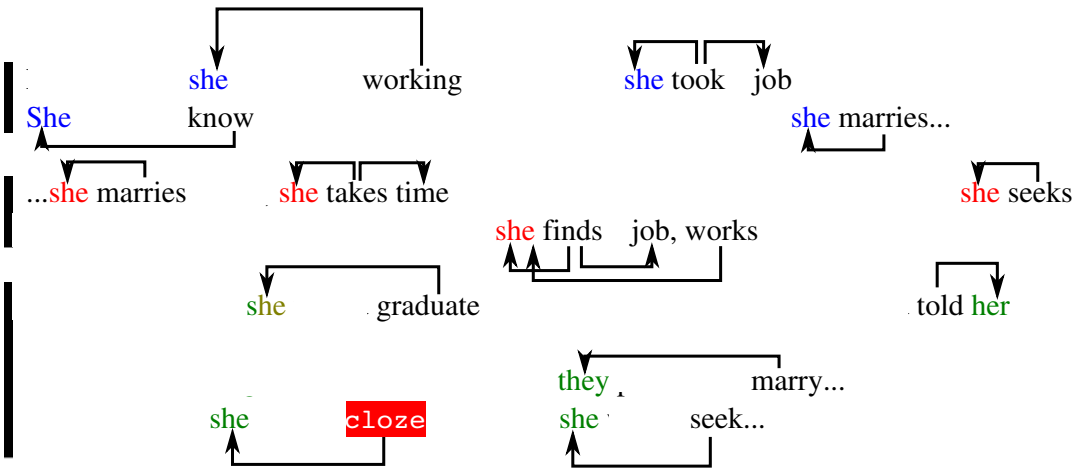


Figure 2.5: An illustration of the cloze task. Each word sharing a color is linked by the same co-reference chain. The arrows indicate a dependency between an event and a linked word. The missing word is covered by the red `cloze` label.

how well they fit in the chain, then save the rank of the correct answer. The reported performance of the system is the average rank—the lower, the better.

Notably, the cloze task does not evaluate narrative schemas, narrative chains, or frames themselves. Rather, it evaluates the model used to make the individual choices that result in schemas.

Originally, the cloze task was done with human beings, wherein individuals would fill in words that had randomly been removed from a sentence (Taylor, 1953).[3] In the context of narrative schemas, random events are removed from chains, and the model used to generate schemas is used to fill in the blanks.

[3]Second-hand citation: Chambers & Jurafsky (2008, 2009)

Recall@N

Jans et al. (2012) employ a new measure for the score in the narrative cloze task called Recall@N. Essentially, it reduces average rank as the score the cloze task (Chambers & Jurafsky, 2008, 2009) to a binary score. If the correct answer c for filling a gap in a script is such that $1 < c < N$, then that test is rounded off at 1; it is 0 otherwise. The values for each gap tested are averaged. This produces scores between 0 and 1, which are somewhat easier to interpret than the open-ended average rank. Some work evaluated on the cloze task have since adopted this measure (Pichotta & Mooney, 2014, 2015; Rudinger, Demberg, et al., 2015).

Critiques

One of the core problems of the cloze task is that, in its conception, it was not intended as an evaluation (Chambers, 2011), but has been picked up by the natural language processing community as *de facto* one. In some sense, the cloze task is a measure of prescience—whether a narrative model can predict events based on those that co-occurred with it. While in some frames, this is reasonable—for example, if there is a vote, it will either be approved or not—news often lacks the sort of predictability that makes this application of frames useful.

Results reported from Chambers and Jurafsky (2008; 2009) were the average rank of the correct answer on a missing verb slot. This can hurt comparability between work, for example, since ranks can fluctuate due to the number of verbs. Jans et al. (2012) propose one solution to this: the Recall@N score, where the number of times the correct answer for a hidden event verb appears in the top N ranks, the verb is counted as a correct match. While this improves the interpretability of the resulting scores, it does not account for rank inflation, deflation due to an decreased or

increased number of verbs. Jans et al. (2012) also introduce the parameter N without a specific determination of an optimal value. They choose 50 as a value and claim that the "results are roughly similar for lower and higher values of N" without any further explanation or data to support their decision of 50 as a value.

Rudinger et al. (2015) demonstrate that the cloze task can be better thought of as a language modeling problem by introducing the use of a Log-Bilinear Language (LBL) model to solve the cloze task. LBL models (Mnih & Hinton, 2007) use context of preceding words to estimate the probability of a word following a given sequence of words. Rudinger et al. (2015)'s particular implementation uses the words preceding each event in a sequence of events to estimate the probability of an event following in sequence. Using this LBL model, they show significant improvements on the cloze task, and given this, provide two possible explanations. The first is that language modeling techniques such as LBL are superior to existing PMI script models. The second possible explanation is that cloze is not a good metric for evaluating the performance of script models.

2.6.2 The Story Cloze Evaluation

Mostafazadeh et al. (2016) offers a new solution to the evaluation problem—a hand-constructed and annotated corpus of stories. This was done entirely through Amazon Mechanical Turk. In a first task, turkers—Amazon Mechanical Turk workers—were paid to write five sentence long stories. Then, in another task, the final sentence of these stories were removed, and turkers were asked to create plausible but false endings for each individual story. Last of all, in a final task, turkers were paid to decide between the contrived false answer and the original story ending. Only stories for which multiple humans could identify the correct final sentence were used in the story cloze task.

Mostafazadeh et al. (2016) tested multiple existing approaches on solving the story cloze task, including the dead simple baseline of "choose the first option" (51.3% accuracy), n-gram overlap (49.4% accuracy), narrative chains from Chambers and Jurafsky (2008) (47.8% - 49.4% accuracy), among others. The best performers were closest word embedding similarity using word2vec embeddings (Mikolov et al., 2013) generated using GenSim (Řehůřek & Sojka, 2010) (53.9% accuracy) and a deep structured semantic model (Huang et al., 2013) (58.5% accuracy). This top scoring approach projects the four context sentences and the fifth solution sentence into the same vector space, using two separate neural networks for each. These networks are trained on letter trigrams, have a hidden layer of size 1000, and an embedding layer of size 300. Cosine similarity is used to compare the fifth sentence's vector with the vector of the context sentences in decision making.

This makes the improvement over the baseline only 7.1%, making this a notably difficult task. Given the selection technique for the task, humans are rated at 100% accuracy.

Fundamentally, Mostafazadeh et al. (2016)'s work represents a return to the original Schankian prospect of endowing an artificial intelligence system of some sort with common sense knowledge about day-to-day events, moving away from news narratives to personal narratives. This is at ends with our own goal—to analyze news narratives.

The synthetic nature of the experiment harks back to projects such as WordNet—an attempt at building pure, fundamental knowledge of words and their relationships—but takes a statistical corpus re-interpretation of this sort of approach. While the presupposition of a rigorously definable and complete lexicon is disposed of, the labor intensive nature remains, requiring a huge volume of stories that must then be considered for generality, filtered, and annotated for reproducibility. 49,255 stories were written to produce a total of 3,744 test instances. To truly capture narrative

knowledge in this way would presumably require many, many more stories. And then, what about fabulae that don't fit into the five sentence mold, or the fabulae that fit into fewer than five sentences? While the story cloze task has some short-term utility, its own labor-intensive and synthetic nature may limit its long-term impact.

2.7 Generating Narrative Schemas

In addition to introducing the cloze task, Chambers and Jurafsky (2008; 2009) spurred interest in generating *narrative schemas*, discrete units that represent commonly reoccurring narratives in text. This section discusses these works.

2.7.1 Chambers' Framework

In this section, I will introduce Chambers' narrative models for event chains and narrative schemas. In many respects, this work constitutes the foundation that this dissertation iterates upon; therefore, this work will be explained in great detail.

Unsupervised Learning of Narrative Event Chains

Chambers and Jurafsky (2008) introduce the cloze task. They also present the first attempt at solving the cloze task, an approach based on the modeling of *narrative event chains*. These chains are based on the assumption of *narrative coherence*, that "...verbs sharing co-referring arguments are semantically connected by virtue of narrative discourse structure." In other words, it's assumed that verbs that share a co-referent are semantically related, and this assumption is demonstrated to hold through the successful extraction of narrative chains. It is hoped that employing such

scripts in a natural language system can provide critical background knowledge, providing the system with expectations about events likely to co-occur with those that have already been witnessed.

Chambers and Jurafsky (2008) demonstrate an algorithm for the extraction of narrative chains—temporally partial-ordered sets of verb-dependency pairs in which a single entity fills one argument slot in each pair. Two examples in Table (2.2), are *Firing of Employee* and *Executive Resigns*. X and W represent a co-referent in the argument slots.

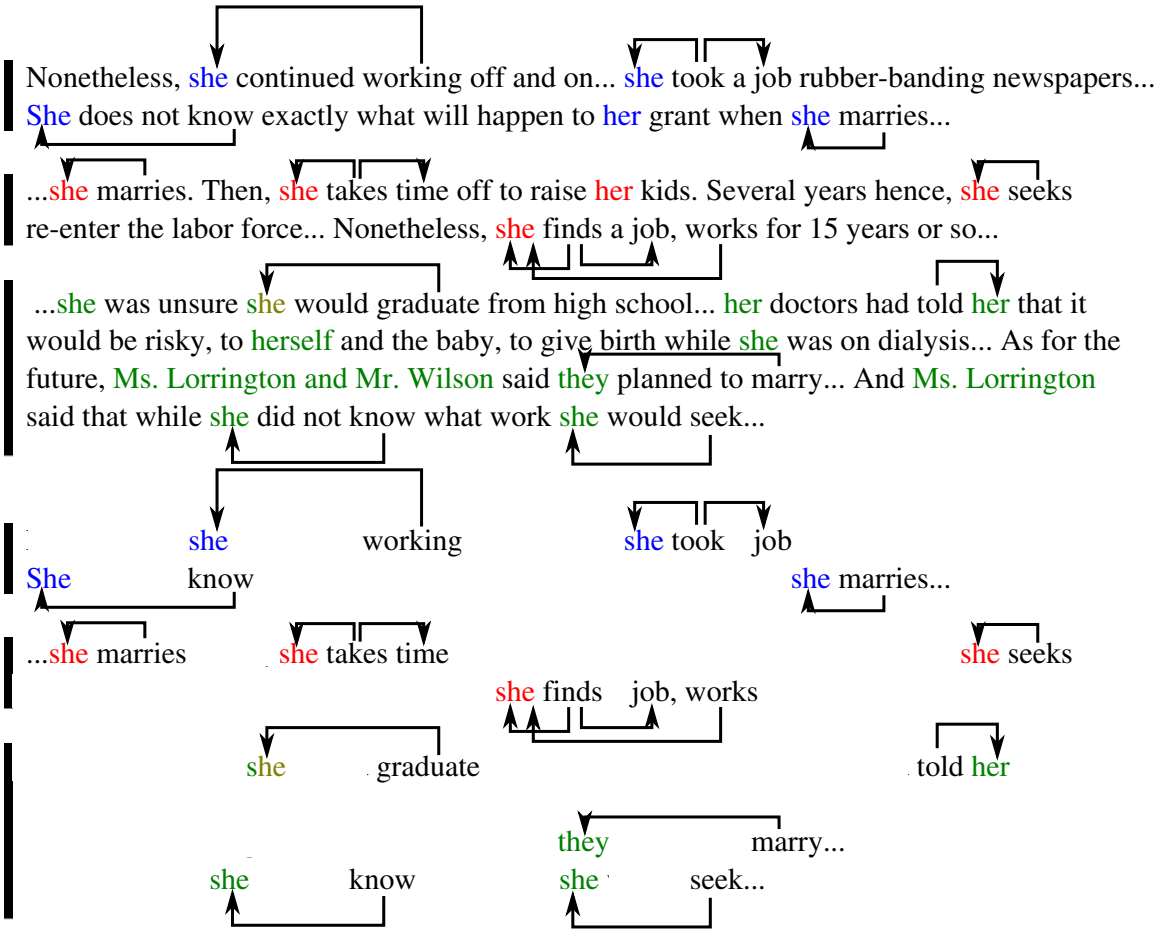
Table 2.2: Examples of two narrative event chains from Chambers & Jurafsky (2008).

Firing of Employee			*Executive Resigns*		
_	accused	X	W	joined	_
X	claimed	_	W	served	_
X	argued		W	oversaw	_
_	dismissed	X	W	resigned	

Notably deviating from their referenced prior work (Mooney & DeJong, 1985), the schemas they build are synthesized from aggregate statistics found in many documents. In Mooney and DeJong (1985), schemas are extracted on a per document basis, each schema being a generalized representation of a handful of similar documents.

At the heart of Chambers and Jurafsky (2008) is point-wise mutual information, often referred to as *pmi*. *pmi* is a more general concept, appearing in a great deal of linguistic work over the last few decades (Manning & Schütze, 1999, Section 2.2.3). Given a probability distribution P, Chambers and Jurafsky (2008) define *pmi* as:

$$pmi(e(w,d),e(v,g)) = \log \frac{P(e(w,d),e(v,g))}{P(e(w,d))P(e(v,g))} \tag{2.2}$$



Document 1: `{<work, SUBJ>, <take, SUBJ>}, {<work, SUBJ>, <know, SUBJ>}, {<work, SUBJ>, <marry, SUBJ>}, {<take, SUBJ>, <know, SUBJ>}, {<take, SUBJ>, <marry, SUBJ>}, {<know, SUBJ>, <marry, SUBJ>}`

Document 2: `{<marry, SUBJ>, <take, SUBJ>}, {<marry, SUBJ>, <seek, SUBJ>}, {<marry, SUBJ>, <find, SUBJ>}, {<marry, SUBJ>, <work, SUBJ>}, {<take, SUBJ>, <seek, SUBJ>}, {<take, SUBJ>, <find, SUBJ>}, {<take, SUBJ>, <work, SUBJ>}, {<seek, SUBJ>, <find, SUBJ>}, {<seek, SUBJ>, <work, SUBJ>}, {<find, SUBJ>, <work, SUBJ>}`

Document 3: `{<graduate, SUBJ>, <tell, PREP>}, {<graduate, SUBJ>, <marry, SUBJ>}, {<graduate, SUBJ>, <know, SUBJ>}, {<graduate, SUBJ>, <seek, SUBJ>}, {<tell, SUBJ>, <marry, SUBJ>}, {<tell, SUBJ>, <know, SUBJ>}, {<tell, SUBJ>, <seek, SUBJ>}, {<marry, SUBJ>, <know, SUBJ>}, {<marry, SUBJ>, <seek, SUBJ>}, {<know, SUBJ>, <seek, SUBJ>}`

Figure 2.6: How Chambers & Jurafsky (2009) count argument slot pairs. First, coreference and parses labeled on segments of three example documents, followed by how the document effectively looks to the counting components of the narrative model. Below these illustrations are the CAPs extracted from each document.

where w and v are verbs, d and g are dependencies. e refers to a *narrative event*—a tuple of a verb—simply referred to as an *event*—and a dependency|some syntactic relation between the verb and some co-referring entity.

Each of these items counted will re-occur in models throughout this dissertation. I will refer to these as *co-referring argument pairs* or *CAPs* for short throughout the dissertation. Fundamentally, *CAPs* are what are extracted from documents in pursuit of schema generation in Chambers and Jurafsky (2008), one of their core innovations. Not all work that follows necessarily uses CAPs, but many have followed their example (Pichotta & Mooney, 2014; Rudinger, Demberg, et al., 2015).

The probability of a CAP is defined as:

$$P(e(w,d),e(v,g)) = \frac{C(e(w,d),e(v,g))}{\sum_{x,y}\sum_{d,f} C(e(x,d),e(y,f))} \tag{2.3}$$

where $C(e(w,d),e(v,g))$ is the number of times a co-reference chain contains some word that has d dependency with verb w and some word that has a g dependency with verb v, modified by a "discount score" (Pantel & Ravichandran, 2004) The numerator is the total number of counts for the CAP of interest; the denominator is the total number of CAPs that were counted at all in the corpus. Figure (2.6) illustrates the NLP pre-processing in full text, how the text "looks" to Chambers' model, then the CAPs that are extracted. These are what are effectively counted by C. If the documents illustrated in Figure (2.6) were the extent of the source corpus, then—ignoring the discount score for this illustration—$C($`{<work, SUBJ>, <marry,SUBJ>}` $) = 2$.

The *pmi* alone is not enough to make chains, however. From the *pmi*, every narrative event chain must be grown *per se*. The most likely verb-dependency pair to be added to a chain is given by Equation (2.4):

$$\max_{j:0<j<m} \sum_{i=0}^{n} pmi(e_i, f_j) \tag{2.4}$$

n is the number of events in the chain being appended; e_i is the i^{th} event in that chain. m is the number of training events in the corpus, and f_j represents the j^{th} pair in the corpus. Effectively, for a particular chain, max searches through every verb-dependency pair in the corpus, yielding the highest total value for the chain. Discrete chains were assembled for illustrative purposes using agglomerative clustering, starting with the *pmi*-based similarity scores between all event types extracted from the data.

How these results are evaluated will be discussed in comparison with Chambers and Jurafsky (2009) to better juxtapose outcomes.

Unsupervised Learning of Narrative Schemas and their Participants

Chambers and Jurafsky (2008) demonstrate how the combination of coreference chains and dependency parses can reveal narrative knowledge. To further refine this knowledge, Chambers and Jurafsky (2009) go beyond the single narrative event chains extracted in Chambers and Jurafsky (2008) and extract whole narrative schemas: intertwined narrative event chains with typed arguments. Typed arguments further inform the scoring of candidate verbs and are learned in the final schemas themselves. This experiment forms the initial foundation of the work developed and presented in this dissertation, and will thus be described in great detail.

In Chambers and Jurafsky (2009), chain similarity from Chambers and Jurafsky (2008) is more formally defined as *chainsim*. For an existing chain C, a candidate verb f and associated grammatical argument g:

$$chainsim(C, \langle f, g \rangle) = \sum_{i=1}^{n} sim(\langle e_i, d_i \rangle, \langle f, g \rangle) \tag{2.5}$$

In other words, the candidate verb and argument are checked against every verb-argument pair already contained in the chain growing in the schema. This is Chambers and Jurafsky (2009)'s reframing of their prior work. They go on to extend this: Formula (2.4) shows the Chambers & Jurafsky (2008) form of *chainsim*, but with $sim(e, e') = pmi(e, e')$.

To include the effect of typed arguments, Chambers and Jurafsky (2009) redefine *sim* as:

$$sim(\langle e, d\rangle, \langle e', d'\rangle, a) = pmi(\langle e, d\rangle, \langle e', d'\rangle) + \lambda \log freq(\langle e, d\rangle, \langle e', d'\rangle, a) \tag{2.6}$$

a represents a specific argument type. $freq(b, b', a)$ returns the corpus count of a filling both b and b'. *sim* contains a superposition of two "fields" of narrative: the verb-argument coreference field and the argument type field. The left-most component $pmi(\langle e, d\rangle, \langle e', d'\rangle)$ represents the reoccurrence of events within the same chains; $\lambda \log freq(\langle e, d\rangle, \langle e', d'\rangle, a)$ represents the reoccurrence of a pair of event verbs with the same argument type. In other words, the first component represents the tendency for the verbs to appear with a common argument regardless of the type of argument they share, normalized by their predicted frequencies if they were independent; the second component counts how often they appear, not normalized by their predicted independent frequencies, with a particular argument.

Though this biases the selection of verbs in favor of adding verbs where an argument type has been shared between a pair of verbs, it does not guarantee that the argument type scores well with the rest of the chain. To do this, we have to check all of the verb pairs in the chain to see how well they all fit together. For this reason, *score* is defined as such:

$$score(C, a) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sim(\langle e_i, d_i\rangle, \langle e_j, d_j\rangle, a) \tag{2.7}$$

This new *score* formula cannot simply be inserted into *chainsim*. To make it compatible with the new definitions, *chainsim*′ is defined as:

$$chainsim'(C, \langle f, g \rangle) = \max_a \left(score(C, a) + \sum_{i=1}^{n} sim(\langle e_i, d_i \rangle, \langle f, g \rangle, a) \right) \tag{2.8}$$

chainsim′ combines the influence of two forces on introducing a new pair $\langle f, g \rangle$ to a chain: how well the new candidate $\langle f, g \rangle$ fits in the existing chain C—determined by the $\sum sim(...)$ component—and how well the argument a fits within the context of the already existing chain C—the effect of the $score(C, a)$ component. In other words, *chainsim*′ looks at how well the new candidate event fits overall possible argument types a and then bonuses the argument types a that work well in the context of the chain being incremented. The types for a given chain are decided by counting the head words of each referent used in each slot of a chain and selecting the most frequent head word for the given chain sequence. Personal pronouns are mapped to a constant `PERSON` type, and other pronouns are ignored.

This extra argument type a is an important addition to the similarity score—it is the core motivating innovation of Chambers and Jurafsky (2009) over prior work (Chambers & Jurafsky, 2008). Because of the $\max_a$, *chainsim*′ wholly contains the benefits of this innovation within its local scope. I employ this score throughout much of this dissertation.

Narrative schemas are defined as a two tuple (E, C). E a set of events—themselves two tuples of v, a verb, and d, a grammatical argument position. Each $v \in E$ is contained in a $c \in C$, the set of sets of typed chains. This structure will form the basis for the sort of schemas discussed later in this dissertation. As mentioned previously, *chainsim*′ is what is evaluated by the cloze task.

A further layer of mathematical legwork is performed to generate schemas from $chainsim'$:

$$\max_{j:0<j<|v|} \sum_{d \in D_v} \max(\beta, \max_{c \in C_N} chainsim'(c, \langle v, d \rangle)) \tag{2.9}$$

In other words, in order of frequency, each verb is either added to a chain of an existing schema C_N or added to a new schema if no existing schema induces a score greater than β for the given verb. The component maximized over is called $narsim$ (Chambers & Jurafsky, 2009), defined specifically as:

$$narsim = \sum_{d \in D_v} \max(\beta, \max_{c \in C_N} chainsim'(c, \langle v, d \rangle)) \tag{2.10}$$

This technique for generating a schema, building from a list of seeds and a score relating those seeds to chains within schemas, will be further discussed in Chapter (3); it is the first example of what I will define as a *germinator* for creating schemas.

Performance

In Chambers and Jurafsky (2009), to demonstrate their algorithm's accumulation of narrative knowledge, schemas and narrative chains are created with increasing large windows of data from the New York Times portion of the Gigaword Corpus. 100 random news articles from 2001 were selected as test documents, a number of which were outside of the subset suitable for generating schemas—that is, they lacked a protagonist with five or more events associated with themselves—leaving 69 articles.

The cloze task itself does not require knowledge of argument types, but Chambers and Jurafsky (2009) show a relative improvement on the narrative cloze task over Chambers and Jurafsky (2008). In the evaluation, they refer to "schemas," but these do not refer to the narrative schemas themselves; rather "schemas" is shorthand here for a model that pay attention to more then only the protagonist chains extracted in Chambers & Jurafsky (2008). In other words, the schemas themselves are not directly

Table 2.3: Percent score improvements when different components are added to the model (Typed Chains and Schemas) and when they are added jointly (Typed Schemas) (Chambers & Jurafsky, 2009).

Model	Improvement
Typed Chains	6.9%
Schemas	3.3%
Typed Schemas	10.1%

evaluated, but the two underlying model improvements—chain types and multi-chain extraction—that make them possible are.

Typed narrative schemas perform better than either untyped chains, typed chains, or schemas without types, with a reported 10.1% improvement in the ranking of verb choices over the untyped chains. Notably, typed chains outperformed both untyped chains and schemas, showing that both arranging chains as schemas and adding type information yields improved results.

2.7.2 Jans et al.

Jans et al. (2012) attempt to address a number of questions with respect to the nature and structure of narrative schemas, namely about representativeness, seeding statistics, and generation. In doing so, they demonstrate an effective implementation of skip-grams to the schema creation process, propose a new measure in the cloze task called "Recall@N" that they argue is more effective than the average rank from Chambers and Jurafsky (2008; 2009), and a new technique for ranking event fits for the gaps of incomplete scripts. They considered all permutations of a set of possible

algorithm choices to evaluate the best possible means for generating schemas with respect to the cloze task.

Skip-grams, and the Implication of Skip-grams

This work has been credited with redefining the cloze task as one that is text-ordered (Mostafazadeh et al., 2016). The original cloze lacked this feature, but Jans et al. (2012) attained improved cloze performance including text-order as part of their prediction model.

In their own words, Jans et al. (2012) mention that they experiment with the first use of *skip-grams* in building narrative schemas. This only implies their primary contribution to the literature: that text order is included in their model, allowing skip-grams to be a relevant and meaningful consideration. In their basic model, events are taken from event chains in the sequence in which they appear, and events are only counted in sequence. So from the paragraph—not from Jans et al. (2012), but synthesized here to illustrate skip-grams—with emphasis on the coreference chain to be discussed:

> Rebels attacked the base Wednesday to retaliate for *recent air strikes*. *The air strikes* aimed to reduce oil production funding rebel forces. Commander John John commented that *the air strikes* succeeded in dismantling production in refineries on the border, and estimated that *they* caused no civilian casualties.

In that paragraph, there are four relevant dependencies with respect to the chain: `< (retaliate, PREP), (aim, SUBJ), (succeed, SUBJ), (cause, SUBJ) >`. In their bigram model, three bigrams are produced|e.g. `< (retaliate, PREP), (aim, SUBJ) >` but not `< (retaliate, PREP), (succeed, SUBJ)>`. In the 1-skip model, `<`

`(retaliate, PREP), (succeed, SUBJ)>` is produced, but `< (retaliate, PREP), (cause, SUBJ)>` is not. The bigram model's output is a subset of the 1-skip model. They also include a 2-skip model, which would include `< (retaliate, PREP), (cause, SUBJ)>` and all of the bigrams from the 1-skip model.

Despite claiming to reproduce what Chambers and Jurafsky (2008; 2009) actually did—count every pair of events in a coreference chain—they do not really. They try to capture this with a formula that forces symmetry into their ordered bigram model, but it still fails to capture counts for all members of a coreference chain. In effect, in terms of their n-gram notation, Chambers and Jurafsky (2008; 2009) employ a symmetric, ∞-skip-gram approach to counting, so their claims of superior performance are really against a straw man. They *do* show, however, that a model that accounts for document ordering obtains better performance than a similar symmetric one. It's not clear that this generalizes to the ∞-skip-gram model.

Results

All results are evaluated with the cloze task. Jans et al. (2012) look at two different corpora: a corpus of fairy tales and the Reuters corpus (Lewis et al., 2004).

With respect to representativeness, Jans et al. (2012) consider different *selection methods* for event chains, and find that the choice of pairs of events that are counted for schema generation affects results differently, depending on the corpus. For the larger Reuters corpus, taking all event chains performs better than techniques that reduce the number of chains. For the fairy tale corpus, however, "long chains" performed best—those with five or more events in a chain. They speculate the performance degrades because the smaller corpus has fewer event tokens to smooth out noise. In both cases, the score differences—from the worst to best performers—were less than 2%.

Statistics are counted in different ways, referred to as the choice of *counting method*. This is how—from extracted coreference chains—the language model is trained. 2-skip grams perform the best over both corpora; 1-skip performs equally on the fairy tales. During their discussion, Jans et al. (2012) claim that this indicates a new finding—however, as discussed above, the way skip-grams are presented here reflects a subset of Chambers and Jurafsky (2008; 2009)'s own n-grams used for counts. The tendency of their improved results with increasing n validates the counting used in prior work.

2.7.3 Generating Coherent Event Schemas at Scale

Balasubramanian et al. (2013) explore a number of variations on Chambers and Jurafsky (2008; 2009). They investigate a wide range of features in an attempt to generate schemas, using a graph-based technique for schema generation, and evaluate via crowdsourced rankings. In some respects, it is an outlier with respect to other work around the time. Nearly all existing follow-ups to Chambers and Jurafsky (2008) include some evaluation using the cloze task. Balasubramanian et al. (2013) do not, which makes it difficult to compare to similar work.

Instead of extracting verb-dependency pairs, tied together with co-referrence chains, they obtain triples of $<$ `arg`$_0$`, relation, arg`$_1$ $>$. `relation` is any of a number of possible relations: a verb or a preposition and verb. `arg`$_0$ and `arg`$_1$ act as role fillers of `relation`. These are referred to as "rel-grams." They provide the example "He cited a new study that was released by UCLA in 2008," which yields three tuples:

1. <He, cited, a new study>

2. <a new study, was released by, UCLA>

3. <a new study, was released in, 2008>

These tuples are further normalized with stemming, regular expressions for dates and personal pronouns, and named entity tags from the Stanford NER (Finkel et al., 2005).[4] These normalized entity types are generalized with hand-selected WordNet 2.1 senses (Fellbaum, 1998). All combinations of normalized role fillers are applied, resulting in four possible tuples per original rel-gram. For example, (1) above normalizes in four different ways:

1a. <He, cite, study>

1b. <He, cite, `activity`>

1c. <`person`, cite, study>

1d. <`person`, cite, `activity`>

From there, their language model approximates the probability of any two tuples co-occurring in sequence, like Jans et al. (2012). P_k estimates the probability of two tuples co-occurring within a window of tuples k, smoothed by δ:

$$P_k(T'|T) = \frac{\#(T, T', k) + \delta}{\sum_{T'' \in V} \#(T, T', k) + \delta \cdot |V|} \tag{2.11}$$

P further reduces the strength of correlations depending on their distance from one another:

$$P(T'|T) = \frac{\sum_{k=1}^{10} a^k P_k(T|T'')}{\sum_{k=1}^{10} a^k} \tag{2.12}$$

$\alpha < 1$ to appropriately approximate the intuition that tuples located further from one another are less related.

[4]Second hand citation: (Balasubramanian et al., 2013)

To generate schemas, Balasubramanian et al. (2013) use a graph-based model. These probabilities that constitute the language model form the edges of a graph of different tuples from which schemas are built. Each edge's weight is the symmetric conditional probability, $P(T|T') \cdot P(T'|T)$. Nodes with high connectivity seed the search. Sub-graphs are extracted beginning with these seed nodes, extending out two hops. A particular variant of Page Rank (Brin & Page, 1998)—Personalized PageRank (Haveliwala, 2002)[5]—ranks items with respect to a particular node, better suiting the seeds in this task.

To build schemas from page rank scores, the top n tuples are extracted. For each pair of arguments, their role fillers are extracted, and the relation between them is added to a list for that pair. If earlier two role fillers were found to be equivalent, they are merged. Then, actors that performed similar actions were merged—that is, if A_1 and A_2 are both connected to A_3 through R, then A_1 and A_2 are merged. Some types are prevented from merging: locations and dates. Additionally, a merge is not allowed if it results in one actor filling both argument positions. These mergers create participants in the schemas without the use of coreference information.

The final result is an ordered list of tuples, using the yielded set of actors and associated relations. These schemas were evaluated using Amazon Mechanical Turkers. Each turker is presented with some combination of role-fillers and events, designed to allow them to identify the following:

1. "Does the schema belong to a single topic?"
2. "Do tuples [in a schema] assert valid real-world relations?"
3. "What proportion of tuples [in a schema] belong?"
4. "Do actors represent a coherent set of arguments?"

[5]Second-hand citation: (Balasubramanian et al., 2013)

For example, they were given the schema with a single role-filler given from the top five best-fitting types for each of the slots, then they were asked if the tuples contained therein were "meaningful in the real world" and if "each tuple [in the schema] belong[s] to a common topic."

Fundamentally, while containing many novel innovations with respect to prior work, it is hard to say what the main takeaway of this work is. Many small tweaks to existing features were altered and improved performance was demonstrated using some of those tweaks, but what fundamentally made Balasubramanian et al. (2013)'s architecture superior is unclear. It represents a solid piece of engineering, but with a scientific contribution that's difficult to pin down.

Even while conducting the first systematic manual evaluation of unsupervised schemas, the merits and successes of the evaluation—especially for the purpose of conducting an unsupervised, critical analysis of news text—are unclear, especially considering the impressionistic interpretations and assumptions expressed by Balasubramanian et al. (2013) in devising their criteria.

For example, should every schema fit into a coherent topic? Could it not be the case that some schemas belong in multiple topics? Do mechanical turkers actually understand what linguists mean by topic?[6] Do linguists understand what linguists mean by topic? These are important linguistic questions that Balasubramanian et al. (2013) largely presupposed answers to.

Even more so, many ungrounded assumptions went into devising their evaluation. Consider, in their own words:

> 'Our instructions specified that the annotators should ignore grammar and focus on whether a tuple may be interpreted as a real world statement. For example, the first tuple in R1 in Table 5 is a valid statement— "a

[6]Balasubramanian et al. (2013) reported no agreement scores with their Turk results.

bomb exploded in a city", but the tuples in C1 "a blast exploded a child", "a child detonated a blast", and "a child planted a blast" don't make sense.'

In light of this, consider Figure (2.7), where a child was armed as a suicide bomber in Mosul. While perhaps metonymic uses, the meaning is very clear, and while the harsh realities of the world might not make sense to Balasubramanian et al. (2013), here they sit before us, a product of the very sort of news used to generate the schemas we discuss.

Figure 2.7: An Iraqi SWAT member disarms an explosive vest wrapped around a twelve-year-old during the Battle of Mosul (AlMalek, 2017). Balasubramanian et al. (2013) used an example of this very sort of real world event as an event that does not "make sense" and cannot be "interpreted as a real world statement."

Chambers' schemas had stumbled upon evidence of this while Balasubramanian et al. (2013)'s guidelines had considered it erroneous. In performing a critical analysis of news text, these details should not be ignored but interpreted.

2.7.4 Reproducing the Restaurant Script

Rudinger et al. (2015) reproduce Chambers and Jurafsky (2008)'s narrative chain technique and apply it to a limited domain corpus called the "Dinners from Hell

corpus," which consists of user-submitted "stories of their terrible restaurant experiences."

Rudinger et al. (2015) use the same techniques roughly as Chambers and Jurafsky (2008), but include in a few variants as well. They consider variants of the atomic elements: skip n-grams (Jans et al., 2012), coref chains spanning more than 5 events (Jans et al., 2012), a minimum threshold for CAP counts (must be attested in 5 documents). They also consider different model types as well, including Jans et al. (2012)'s ordered *pmi* model, and a bigram probability model. Discounting is used for both the *pmi* models and absolute discounting for the bigram model. There is also a penalty for events that appeared in fewer than D documents, though this parameter is undefined.

Rudinger et al. (2015) used a variant of the narrative cloze task constrained by a hand-annotation of their source corpus for restaurant-related event verbs. Their implementation of the cloze task is limited to verbs that were deemed restaurant-related by their annotators. The baseline was the unigram model from Pichotta and Mooney (2014), the verbs simply ranked by frequency. Overall, all methods that were tried exceeded the baseline technique under with respect to the average rank of each model on the cloze task. With respect to Recall@N, however, no model exceeded the baseline.

Nevertheless, they demonstrate successful application of existing script-methods to smaller, topic-specific corpora, given that their own corpus was a mere 143 stories that were an average 352 words each.

2.8 Predicting Event Verbs

Following Chambers and Jurafsky (2009), a number of approaches to solve the narrative cloze task were introduced. These typically do not seek to produce schemas as an end product but instead seek to optimize cloze performance, often using a generative model to do so.

2.8.1 Statistical Script Learning with Multi-Argument Events

Pichotta and Mooney (2014)—in a similar vein to Balasubramanian et al. (2013)—learn script models with multiple arguments. That is, their model of events is learned as a tuple including an event and its arguments jointly. In Chambers and Jurafsky (2008; 2009), pairings of subject and object are only learned implicitly through common pairings of events. In Pichotta and Mooney (2014)'s model, these pairings are explicitly remembered in the model being learned as whole tuples.[7]

Pichotta and Mooney (2014) learn multi-argument events, like Balasubramanian et al. (2013), but with a more constrained generalization process and through leveraging coreference information. Instead of counting "rel-grams" of all possible combinations of types and generalizations of types, Pichotta and Mooney (2014) use coreference information between two entities to collapse different slots amongst different events together, with slots that have no shared coreferents mapped to a fourth O set.

Given that event ordering is included in Pichotta & Mooney (2014)'s model, the baselines used follow from that. The *random baseline* selects verbs at random from the observation set. The *unigram model* guesses event verbs based on the probability

[7] I have spelled this out explicitly since a reviewer previously was confused on this point. `PREP` was included in Chambers and Jurafsky (2008; 2009)'s original work. Balasubramanian et al. (2013) preceeded Pichotta and Mooney (2014) in this style of mutli-argument tuples as well with their "rel-grams."

of the preceding verb. The *single protagonist model* uses CAPs to chose the event that maximize the probability of the selected event verb, the same as Jans et al. (2012). The *multiple protagonist model* extends the single protagonist model in a straight-forward way to handle the type of multi-argument events used in their primary model.

Pichotta and Mooney (2014) do not have any sort of germinator since they do not use their model to generate schemas of any kind.

In evaluation, they use Jans et al. (2012)'s Recall@10 and a new accuracy score—the accuracy reflects the accuracy of both the event choice and the accuracy of the other slot selections On the random baseline, R@10 was 0.001 and accuracy was 0.334 for verbs that have three argument slots, reflecting that there was a great chance that at least one of the arguments would have been selected correctly at random. Their best performer, a model that learns pairs of events using multi-slot arguments, achieved a R@10 of 0.336 and an accuracy of 0.561.

2.8.2 Generative Frame Models

Cheung et al. (2013) started a thread of work centered around creating generative frame models. These were often evaluated using MUC templates. Cheung et al. (2013) present a frame prediction system that is more formally grounded probabilistically than its predecessors, producing a generative model, mapping a set of frames over a document that maximizes the probability of the document given the model. They claim that:

> "Perhaps the most similar to our frame is Roger Schank's scripts, which capture prototypical events and participants in a scenario such as restaurant dining."

This claim is not quite accurate for a number of reasons—namely that they model interactions between frames. Scripts themselves are discrete units (Schank & Abelson, 1977). Modeling frames is inherently Fillmorean; adding probabilistic transitions between them does not cross the line into Schankian. Nevertheless, the work itself is related—if only via the similarities enumerated—and worth discussing.

Formally, the goal is to find the set of frame assignments that maximize the likelihood of a given document set $\mathcal{D}$. The probability of $\mathcal{D}$ is estimated with the `ProFinder` model which uses two hidden markov models: one frame related and one event related. The frame HMM emits events and has underlying state transitions between frames; the event HMM emits argument slots and has state transitions between events within a frame.

No coreference information is used or carried between these transitions, so the model lacks a persistent knowledge of entities through the discourse. Consequentially, such a model makes no distinctions between "A chases B. A kills B" and "A chases B. B kills A." These constitute very different stories with very different outcomes, but a model lacking coreference may only incidentally be able to comprehend or infer the differences between both such stories. This effectively makes it more a model of language rather than a model of the knowledge perceived and retained in comprehending discourse.

The model has a few adjustments deviating from a theoretically pure model into a functionally effective practice. A special background frame captures the insertion of generic content, such as speaker attribution of quotes, that is likely to appear in any frame. Also, adjustments were made to some of the transition probability estimates to account for the "stickiness" of frames—that is, the tendency for frames to remain the same in discourse.

Results were evaluated through two different tasks—MUC-4 (United States Defense Advanced Research Projects Agency. Software and Intelligent Systems Technology Office, 1992) and TAC 2010 (Owczarzak & Dang, 2010)—cross-compared with results using templates extracted by Chambers and Jurafsky (2011). Their own frames had higher F-scores on both tasks, but Chambers and Jurafsky (2011)'s templates retained a higher precision score in both tasks, likely because they were smaller in size (Cheung et al., 2013). No evaluation on the cloze task was performed.

Following Cheung et al. (2013), Chambers (2013) includes a number of innovations in schema induction, many of them juxtaposed with discoveries found in Cheung et al. (2013). Both works fall into the category of a "generative" framework. While Cheung et al. (2013) create a model of frames over a document, Chambers (2013) overtly creates schemas from that generative model. In differentiating itself from Cheung et al. (2013), Chambers (2013) employs schemas that are entity-driven as opposed to sequentially-driven—that is, Chambers (2013) employs entity coreference to drive schema generation, whereas Cheung et al. (2013) only employs the local set of possibilities learned from an event.

Chambers (2013)'s entity-driven approach demonstrably performs better than Cheung et al. (2013)'s and Chambers & Jurafsky (2011). They note that the algorithm, as it's employed here, is the same for both learning and extraction. It implicitly conducts the task of document filtering—something many MUC-4 models require before hand, resulting in a linear extraction process or "pipeline." This model of schema extraction does not have such a constraint. While being entity-driven, Chambers (2013) lacks the sequential knowledge of Cheung et al. (2013)—they point out that there is room for investigation in the space combining these two.

Nguyen et al. (2015) extend the generative, entity-driven approach of Chambers (2013) further. In addition to using the head of noun phrases as a means of typing

entities, they also include *attributes*—other nominal, adjectival, or verbal components of noun phrases that prior work glossed over. For some reason, they chose to call events *triggers*, but this lacks any clear explanation as to why they do not simply call these "events."

Nguyen et al. (2015) also evaluated on the MUC-4 corpus, experimenting with and without attributes, as well as with and without coreference information. Attributes showed some very minor improvements in precision, recall, and F1 scores, at best improving F1 score by around 1% across models. Coreference information showed greater improvements, improving F1 scores by 2 - 3%.

2.8.3 Deep Network Frame Models

Two systems have been devised to date that leverage recent developments in recursive neural networks (Mostafazadeh et al., 2016; Pichotta & Mooney, 2015). These are discussed in this section. While both frameworks leverage recent developments in recurrent neural networks to create some sort of model of script knowledge, they both use different evaluations to determine the quality of their systems, so finding comparisons between the two is difficult. Additionally, Mostafazadeh et al. (2016)'s model is a brief mention in their work where they use an existing model out of the box to demonstrate their new task (Huang et al., 2013), in addition to a number of other baselines. Pichotta and Mooney (2015; 2016), on the other hand, focus more specifically on their architecture.

Pichotta and Mooney (2015, 2016)'s LSTM Models

Pichotta and Mooney (2015) demonstrate a Long Short-Term Memory (LSTM, see Hochreiter & Schmidhuber (1997)) model for script learning. This is done through

two different tasks: the narrative cloze task (Pichotta & Mooney, 2015) and the text-prediction task (Pichotta & Mooney, 2016).

The model in Pichotta and Mooney (2015) employs underlyingly co-referring argument tuples to represent events, most similar to those in Pichotta and Mooney (2014) but including the preposition itself in the tuple, resulting in 5-tuples instead of 4-tuples. Sequences of these tuples are fed into the LSTM, implemented as described in Zaremba & Sutskever (2014).

The LSTM itself is a more sophisticated implementation of a recursive neural network that allows for the network to more easily learn long-distance dependencies, a property which Pichotta and Mooney (2015) argue makes them wholly appropriate for script learning. The memory units of the LSTM contain functions that are easily differentiable, allowing for "standard gradient-based methods" to train the network. The network is trained on sequence of event inputs, with each item in sequence acting as input to predict the next item in the training sequence. The system fundamentally receives three inputs: w_t, a 1-of-V vector indicating the word at t, where V is the size of the vocabulary; c_t, an vector indicating the event component being input, e.g. subject, preposition, verb, etc.; and e_t, an entity id for nominal arguments. The output is a prediction of w_{t+1}, the next word in the sequence.

Two types of models are devised for the system. Both models learn to predict verb lemmas and prepositions, but *noun models* predict noun lemmas only while *entity models* predict coreference between events. These two models of the data result in four different architectures for input and output. Two architectures model inputs to predict a similar type of output: e.g. noun-noun and ent-ent; two other architectures predict either noun lemmas or entity connections using both lemmas and prior entity connections.

Four baselines are used in Pichotta and Mooney (2015). Two, the *unigram* and *bigram* models, are similar to those laid out in Pichotta & Mooney (2014), but adapted to the new task. The *re-written bigram* baseline works roughly the same as the bigram baseline except that it also "hallucinates [sic]" any co-occurring events with all possible arguments to those events; the *2D re-written bigram* model works the same as the re-written, except that its objective function is the same as that described in Pichotta & Mooney (2014), representing the best performing published system on the cloze task.

English Language Wikipedia is used as the source data, resulting in 8.9 million event sequences. Within tuples, the top 50 prepositions, 2,000 verbs, and 8,000 nouns are used in the vocabulary; all others are reduced to OOV items.

Two quantitative tasks were used to evaluate the resulting event predictions: the cloze task, with scores softened both by recall@25 and with an average WordNet proximity score (Fellbaum, 1998; Wu & Palmer, 1994). All four LSTM models show improvement over prior systems in all circumstances, using Recall@25 (Jans et al., 2012) and an accuracy score that softens mismatches using a WordNet based similarity score. The greatest improvement was 64.9% over the best performing baseline with an 18.2% improvement in partial credit accuracy.

Furthermore, Pichotta & Mooney (2016) deploy the Pichotta & Mooney (2015) system to make predictions of raw text—that is, using a sentence in sequence to predict the content of the next sentence. The system is evaluated on holdout documents and given a Bleu score on its ability to make predictions. It was found that learning a direct text-to-text encoding performed better at both predicting text and predicting events than encoding events-to-events like in Pichotta & Mooney (2015). However, as noted,

> "In open-domain text, a sentence is typically not straightforwardly predictable from preceding text; if it were, it would likely not be stated."

This pragmatic problem is as much a fault of the cloze task as it is a problem of Pichotta & Mooney (2016)'s new task, carrying along cloze's faults with it, while not offering a new linguistic or theoretical framework to better understand the nature of the scripts.

In fact, while reasonably successful at achieving its benchmark, this work fundamentally begs the question as to whether this is even script knowledge. If we define script knowledge as a model of tendencies, then the argument can be made that text prediction indeed embodies script knowledge. However, if a script is defined as a discrete structure and if discrete structures are helpful and necessary for downstream tasks, then the answer is no. There is nowhere in such a model to provide an inventory of a discrete set of scripts. In the words of Schank and Abelson (1977):

> "If one falls back on the abstract position that only form is important, that the human mind is capable of developing knowledge structures of infinitely varied content, then one sacrifices the essence of the structure concept, name the strong expectations which make reality understandable. In other words, a knowledge structure theory must make a commitment to particular content schemas."

In other words, from their position, what makes the schema or script concept valid is its ability to put us discretely and absolutely into a particular category of activity. A model that allows indefinite variation—such as through a probabilistic model—at no point makes such decisions. Of course, it may be the case that Schank and Abelson's intuition about this is wrong, but there is definitely an incompatibility between both of these modes of thought about knowledge.

Mostafazadeh et al. (2016)

Mostafazadeh et al. (2016) presented a number of possible solutions to their proposed story cloze task. An LSTM-based model—the Deep Structured Semantic Model (or DSSM) (Huang et al., 2013)—performed best with 58.5% accuracy over an "Constant-choose-first" baseline of 51.3%. This model attempts to project the four context sentences and the solution sentence into the same vector space. It does so with two separate neural networks, one for the context sentences and one for the solution sentences. The embeddings themselves are 300-dimensional and are produced with a size 1000 hidden layer. The DSSM takes as input "context dependent characters," that is "letter-trigrams" of the source material. The solution sentence that's chosen has the highest cosine similarity between itself and the context vector.

This has since been improved in the LSDSem 2017 workshop where improvements were demonstrated, up to 75.2% (Mostafazadeh et al., 2017). The `msap` model from the University of Washington used logistic regression with a number of features helpful in authorship identification, NLTK tokenization, and POS tags from SpaCy (Schwartz et al., 2017). Notably, they used no word embeddings whatsoever in their solution, beating many different approaches that did.

2.9 Discussion

A few themes have emerged through this literature review which I discuss in this section. These include the sorts of components used by different script inference systems, the cloze task, and the underutilization of narrative schemas as a tool for text analysis.

2.9.1 Looking Back (and Forward) at the Cloze Task

The cloze task has been largely responsible for spurring renewed interest in script models, and in itself, has a number of advantages. It is easy to deploy—ranking candidates by score affinity to a particular event chain is trivial to reproduce. It also does not require a specialized annotation or corpus to evaluate on. Rather, only a set of held-out documents from any corpus is sufficient for the task. This made it easy for follow-up work to deploy their own models, leading to the cloze task's ubiquity.

However, the cloze task is not without faults. Rudinger, Rastogi, et al. (2015) and Mostafazadeh et al. (2016) both give distinct critiques of the cloze task. Mostafazadeh et al. (2016) consider the cloze task one that lacks any day-to-day, "commonsense knowledge," with much of the existing work optimizing around the shallowness of the problem. On the other hand, Rudinger, Rastogi, et al. (2015) critique the cloze task on the grounds that LBL learners perform quite well on it, meaning that LBL learners are good script learners or that the cloze task is not an evaluation of script knowledge. While Rudinger, Rastogi, et al. (2015) do not outright reject that LBL models are good learners of script knowledge, their reliance on more general stylistic content available to the reader suggests that cloze solutions are better guessed with linguistic style rather than world knowledge. Jans et al. (2012) in some respects suggests this at well, exploiting the text order to make better guesses rather than any concept or actual sense of world ordering—this too is a more stylistic solution rather than knowledge base one.

I would like to take these critiques a step further. Rather, the cloze task is a measure of *prescience*. Many narratives have an infinitude of possible and valid continuations from event-to-event. Even for a human being to predict these continuations accurately is impossible in many circumstances. Whether or not we are doing

an ordered or unordered version of the task, it will in many circumstances require our system to predict an event for which the surrounding events provide no conventionalizable or commonsense information. Pragmatically, the very motivation for discussing an event is the fact that it is novel, and if conventionalized components appeared with it, then they would be redundant.

In any case, it is clear that other evaluations could be helpful, especially some that do not require a specialized corpus. In some sense, we have to take into account the original theoretical considerations from which scripts were derived (Schank & Abelson, 1977) but provide a more robust interpretation of those considerations than the cloze task enables.

Nevertheless, since the cloze task is so easy to deploy, it is likely not going away. However, new evaluations may corroborate its decisions or show its obsolesce.

2.9.2 The Underutilization of Narrative Schemas

While many models have been proposed for capturing the connections between events in text, few have gone beyond the narrative cloze task, and while capable, few have been used to actually generate narrative schemas. This begs a lot of questions about schemas themselves. What can they be used for? Are they actual linguistic objects or mere artifacts of a quantitative process? Can they lend insight into understanding differences in narrative and discourse structure across corpora?

The existing literature really doesn't delve into these questions well. No known work has used a script model for performing a task outside of those used to evaluate script models, or to conduct an unsupervised investigation into the distributional properties of schemas in text, or even to investigate the properties of schemas: for example, their their distribution in text, their interactions with document categories, or their sensitivity to changes in their source data.

2.10 Conclusions

In this literature review, I discussed existing work on narrative schemas, models of script and schema knowledge, and their theoretical underpinnings. In the context of this, I discussed ways in which such models are evaluated and implemented. Overall, while script models have come into focus, schemas have fallen by the wayside. In the next chapter, I will look back at Chambers and Jurafsky (2009) and the schemas they generated, presenting new ways to create such schemas and compare qualitatively the products of these different ways of generating schemas.

Chapter 3

Generating Schemas

3.1 Introduction

As discussed in the previous chapter, in most studies, generating schemas has fallen to the wayside in favor of focusing on improving performance on the cloze task. However, I will begin the analysis of this dissertation by considering different techniques for generating schemas and showing that, while the underlying model obtains the same performance on the cloze task, the schemas generated are objectively different from one another.

To do so, I begin by defining schema *germinators*, the component of the schema generation process that has been skipped throughout most of the existing literature on frame models. I describe this separation between germinator components and narrative/frame language models in Section (3.2) and describe Chambers and Jurafsky (2009)'s own work partially in these terms (Section 3.3). I will describe an existing model in these terms (Chambers & Jurafsky, 2009) and propose two new germinators for schema generation: counter-training (Section 3.4) and a random walker (Section 3.5). All three will be compared qualitatively (Section 3.9).

3.2 Germinators

Fundamentally, the cloze task evaluates the fitness of the *score* used to determine the fitness of a candidate to a chain's event slots (Chambers, 2011). For example, in

the case of Chambers and Jurafsky (2009), this score was *chainsim′*. While the vast majority of prior work focused on evaluating these such scores, little effort has been put into evaluating what comes beyond them, such as schemas. For example, in Chambers and Jurafsky (2009), a number of further steps are taken involving *chainsim′* to generate discrete schemas, but the product of these steps is never evaluated. I will refer to these steps, processes of going from a cloze-evaluable score to a set of narrative schemas, as schema *germinators*.

Germinators contain a number of choices that grow single events into discrete schemas. For example, how are new schemas started? What verbs are considered to add to schemas and when are they considered? How are scores for new candidates for chains interpreted?

In the following three sections, three different germinators will be described. They all rely on the same score, *chainsim′* (Chambers & Jurafsky, 2009). The results presented in this chapter show that different germinators can produce wildly different results despite beginning with the same score.

3.3 Chambers' Linear Induction (LI) Germinator

In this section, I describe Chambers and Jurafsky (2009)'s schema generation germinator, as independent from the score they use, *chainsim′*. Loosely speaking, Chambers' germinator is *narsim* plus the linear way in which verbs are introduced to it. This is why I refer to it as Chambers' *linear induction* germinator.

The schema insertion procedure, here denoted schema insert(...) and shared across all germinators in this dissertation, is specified in Section (3.6).

Linear induction begins with a list of candidate event verbs, ordered from most to least frequent. Each candidate is considered against an existing set of schemas. If the

Algorithm 1: Variant on Chambers and Jurafsky (2009)'s germinator for narrative schema construction. Compared with the prior work, this allows events to appear in multiple schemas when possible.

```
Data: list of event verb candidates, a scoring function scoring, β, max schema
      size M
Result: narrative schemas
grownSchemas = [];
for candidate in candidates do
    noInductionsFlag = True;
    for schema ∈ schemas do
        if scoring(schema, candidate) > β then
            schema insert(schema, candidate);
            noInductionsFlag = False;
            if |schema| > M then
                schemas -= {schema};
                grownSchemas.append(schema);
    if noInductionsFlag then
        schemas.append(new schema([candidate]))
return grownSchemas + schemas
```

scores are too low—less than the β parameter for all schemas under consideration—a new schema is started with the candidate verb. If the schema under consideration has more verbs than parameter M, it is removed from consideration when adding future verbs. This is repeated for every candidate verb until no verbs remain.

This version of the germinator contains one variant from the original: that a verb type may be added as an event to multiple schemas. In a purely Schankian sense, an event should belong in a single schema. However, a single verb does not map to a single event, and a verb might represent different sorts of events. If a preponderance of evidence suggests that a given verb type should belong in multiple schemas, then there's no reason to force the algorithm to simply choose the best fit. Addi-

tionally, the germinators presented in Sections (3.4 and 3.5) also allow for a given verb type to appear as events in multiple schemas, so this improves the comparability between them and linear induction. Given the nature of the linear induction germinator, though, only a handful of schemas are considered simultaneously, so removing this hard constraint has little effect compared to the original. This will be illustrated in Section (3.9).

3.4 Counter-training (CT) Germinator

In this section, I describe a germinator that generates schemas based on a global—rather than local—maximization of similarity. Chambers and Jurafsky (2009) descend the list of verbs, one at a time, through all verbs, only considering one verb and a handful of available schemas to add the verb to. As a result, it is not entirely clear that the linear induction germinator generates schemas that are globally optimal and best represent the narratives exhibited in the corpus.

The aim is to avoid the creation of schemas resulting from the addition of a verb that only coincidentally showed up at the time the schema was being induced and instead to add events to schemas that fit well with respect to all possible events. However, one of the dangers of a global optimization is that all of the schemas resulting from the process will all converge to schemas containing words that have a strong general affinity. Thus, we want to include forces in our germination process that push back against redundancy.

Yangarber (2003) provides a useful analogy in his description of *counter-training* in the discovery of patterns for information extraction. He notes that an "unsupervised algorithm does not know when to stop learning," so that "in the absence of a good stopping criterion, the resulting list of patterns must be manually reviewed."

Yangarber's algorithm relies on competition among several different learners, each seeking patterns for a different "scenario" (a topic or domain). A pattern might have evidence favoring a learner to select it, but if learners for other scenarios also find evidence to acquire it, that counts against the first learners evidence.

The analogy that carries over to narrative schemas is that they ought to reflect unique sets of events, like Yangarber's scenarios, only allowing for polysemous verbs to represent events in multiple schemas if a preponderance of evidence is provided. Verbs that are more general should be penalized for their affinity to fit into multiple schemas. Whereas with Yangarber's work, various patterns indicative of specific topics were the topic of competition, individual schemas should compete for candidate verbs, which are thus the analogs of patterns.

Each schema ranks potential new additions in competition with other schemas. The specific process for this is detailed in Algorithm (**??**). The schema insertion procedure, here denoted schema insert(...) and shared across all germinators in this dissertation, is specified in Section (3.6).

Counter-training schemas requires three parameters: a set of seed schemas, a score, and a pruning condition. In this case, the seed schemas each contain one of the top 800 verbs, thereby setting the number of schemas to 800. The score is $chainsim'$, and the pruning condition for schemas is simply that a schema contains six or more events, plus a special case that is built in to the germinator—candidates for a specific schema are pruned if they have a score <= 0.0 after penalties or have already been added to a schema, and if a schema has no induction candidates, its growth is terminated. While the algorithm allows for more sophisticated pruning conditions, this has been kept simple in this case for speed and comparability across algorithms—Chambers and Jurafsky (2009) experimented with different schema sizes but only used schema size as a pruning condition.

Algorithm 2: Counter-training for narrative event chain construction.

Data: a list of *SchemasGrowing* (schemas containing only one event), a list of event verb *candidates*, a scoring function *scoring*, schema termination condition *term* ($\lambda s : |s_e| <= 6$)

Result: narrative schemas

while *len(SchemasGrowing)* > 1 *and len(candidates)* > 1 **do**

    simtables = [];

    for $S \in$ *SchemasGrowing* **do**

        CandidateTable = {};

        for *candidate* $\in$ *candidates* **do**

            CandidateTable[candidate] = $scoring(S, candidate)$;

        simtables.append(CandidateTable);

    broadness = {} **for** *candidate* $\in$ *candidates* **do**

        broadness[candidate] = $\sum_{s \in S} \sum_{c \in s} 1$;

    for *simtable in simtables* **do**

        for *candidate* $\in$ *broadness* **do**

            $simtable[can] -= broadness[candidate]$;

    schema insert(S, argmax$_{can \in candidates} simtable[can]$);

    for $S \in$ *SchemasGrowing* **do**

        if $term(S)$ **then**

            SchemasGrowing -= {S} GrownSchemas.append(S)

    for *candidate* $\in$ *candidates* **do**

        if $simtable[candidate] < 0$ **then**

            candidates -= candidate

return *GrownSchemas*

At every iteration of growth, there are three steps. First, all candidates for each schema are scored using the provided score. These values are gathered into the *simtable*. Next, using the *simtable*, the *broadness* of each candidate is computed. The *broadness* for each candidate event is the sum of the scores across all schemas—this is not the score itself. Finally, each score for each candidate in the *simtable* is penalized based on the broadness, and the highest-ranked candidates are inducted into their respective schemas. The list of schemas and candidates are pruned according to the provided pruning conditions, and the process continues while there are both still candidates and schemas available. Schemas also have a special pruning case—if there are no candidates remaining for that schema, the schema is pruned regardless of whether it has reached the schema pruning condition.

The broadness table allows for schemas to compete with one another, and to do so irrespective of the order they are in. If many competing schemas rank a candidate event highly, they may only add it to themselves if the score outweighs the allotted penalties. This is accomplished through the *broadness* table, which, after adding up how a candidate verb scores against all schemas, is immediately used to penalize the candidate scores in *simtables*. If too many instances of a verb and its dependents seem to fit in different schemas, it is penalized to the point where it is pruned. This does not preclude a verb belonging to two or more narrative schemas, since its individual occurrences might unmistakably belong to one schema or another, even after penalties have been deducted. This is viewed as a positive attribute of the algorithm—if an event has such an affinity for the candidate schema that, even after penalties have been applied, it remains the number one candidate for that schema, then it is difficult to claim that the event under consideration doesn't belong there.

One schema generated through the counter-training process can be seen in Figure (3.1). Shared symbols and colors (e.g. red square) indicate a shared argument slot.

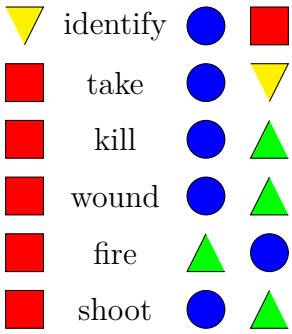


Figure 3.1: A schema extracted using the counter-training technique. The red square and blue circle both indicate different `PERSON`s. The downward pointing yellow triangle indicates some `THINGY`, a fall-back case in the process chain typing process explained in Section (3.8.1); the upward pointing green triangle indicates either `baghdad` or a `THINGY`.

Here, we see a generalization of descriptions of shooting incidents. Some features are worth noting here. The `PERSON` who was *shot* is also likely to have been *wounded* or *killed*. Someone may *fire* bullets *at* someone and also *shoot* someone *with* bullets, reflected in the chains changing places between the `PREP` and `OBJ` slots. Someone *killed* or *wounded* may be *taken* to a hospital, where later the hospital *identifies* the victim.

It is worth noting that the possibility still exists of duplicate schemas themselves still being instantiated during a global maximization process. The schema shown in Figure (3.1) is one such instance, appearing many times with a different event or two rotating out of the slot that *identify* now occupies. While counter-training attempts to push back against this, it's not guaranteed that it will succeed. When such duplication appears, it appears despite resistance supplied by counter-training's penalties, which tells us something potentially interesting about the data. For example, two schemas can easily converge if they were seeded with verbs that were closely related;

once they include the same events, they are effectively identical. This happens occasionally. Specific cases will be discussed in more detail in Section (3.9). And note that while some schemas are redundant here, having overcome the forces pushing against them, they are nowhere near as redundant as those generated by the random walker germinator (Section 3.5), which has no such tendency to push back against redundant schemas.

Note that the counter-training algorithm does not overtly exclude the seed verb of a schema from being induced into the schema again. This is counted as a second verb by the algorithm—in turn, when this duplication is suppressed, the schema appears to be one shorter than the actual verb output.

3.5 Random Walk (RW) Germinator

While the counter-training algorithm is slow, it provides some features that are appealing. Events can appear in multiple schemas, and some schemas can be generated multiple times—an implicit weighting reflecting their frequency in text. To try to retain these features, another way to think of the *pmi* model is as a graph with:

- each node representing an event-dependency pair,
- every weighted edge represents the strength of association between both event-dependency pairs (e.g. the score for a CAP).

Searching the entire graph, as shown effectively through counter-training, is slow Using something like a random walk offers one way of traversing these nodes that's faster and still generates schemas representative of the narrative structure of the source corpus.

The details of the random walker are in Algorithm (3). The schema insertion procedure, here denoted schema insert(...) and shared across all germinators in this

dissertation, is specified in Section (3.6). Because of the interdependencies of content contained in schemas, it varies from a simple random walk, since every step that is made alters the weights on the graph—in other words, the current state of the graph traversal is not enough to compute the relationship between a chain and its candidate verb additions, the nodes traversed previously affect the outbound weights. Specifically, with respect to schemas, the events and chains contained in the schema currently determine the weights to other candidate events to add to the schema. This means that every time an event is added to each schema being generated, all of the weights to new candidate events for that specific schema must be recomputed.

While, like previous models, I use the *pmi*-based $chainsim'$ to weight candidate inductions, the random walker works by sampling the data probabilistically. For this to work properly, the result must be taken back out of the log-based *pmi* space and back into a space more akin to the ratios of the original probability distributions. This is done by exponentiating the score, e.g.:

$$weight(C, vd) = 2^{chainsim'(C,vd)} \tag{3.1}$$

This "undoes" the log from the *pmi* buried deep in $chainsim'$, returning the weights to ratios of probabilities. Without this, rare items are selected too frequently.

The seeder here is the same as the counter-training seeder. It returns a schema containing only one event, with each event represented by one of the most frequent verbs encountered in the corpus. The seeder is implemented as a Python generator. However, since schemas are each generated one at a time by the random walker, they are given to the random walker as needed. Counter-training, on the other hand, quickly seeds all of its schemas for training in advance of beginning germination so that it can begin the counter-training process.

Algorithm 3: A technique for generating narrative schemas with a random walker, weighted by the output of the scoring function.

Data: schema *seeds* (schemas containing only one event), schema termination condition *term* ($\lambda s : |s_e| <= 6$), a weighted score function *weight*, a function *random* that returns random numbers, a list of candidate events *allCandidates*

Result: narrative schemas

```
schemas = [] ;
for schema ∈ seeds do
    candidates = copy(allCandidates) ;
    while term(schema) do
        generate edge weights for all candidates
        simtables = [];
        for candidate ∈ candidates do
            simtables[candidate] = weight(schema, candidate);
        make weighted random choice
        threshold = random() × ∑_{c∈candidates} c;
        weightAccumulation = 0.0;
        for candidate ∈ candidates do
            weightAccumulation += simtables[candidate];
            if weightAccumulation > threshold then
                choice = candidate;
                break;
        candidates -= {candidate};
        schema insert(schema, candidate);
    schemas.append(schema);
return schemas;
```

3.6 Insertion of an Event Into a Schema

In this section, I describe the shared procedure each germinator uses to add a new event to a schema. This does not change between germinators, though hypothetically could.

This procedure is closely related to Chambers and Jurafsky (2009)'s own description of this process, though it at times is vague. "schemas are now learned by adding events that maximize equation 5," referring to maximizations over $narsim$. Implicit in this, this process of adding an event is one of linking slots from the new event to each chain. Particularly, a new slot from an added event verb is linked to the chain which scores the highest on $chainsim'$. If no $chainsim'$ for a new slot is greater than β with any chain, then a new chain is started only containing that slot.

Algorithm 4: Algorithm for adding an event to a schema.

Data: a set of *verbdeps* to insert, a set of *chains* of a schema
Result: a narrative schema
for *verb dependency tuple* $vd \in verbdeps$ **do**
    i = $_{i \in enumerate(chains)} score(chains[i], vd)$;
    score = $\max_{i \in enumerate(chains)} score(chains[i], vd)$;
    if $score > \beta \times len(chains[i])$ **then**
        chains[i].append(vd) ;
    else
        chains.append(([vd], []));

Note that β is the same β used in linear induction. However, while counter-training and the random walker lack such a parameter, they still use it at this step of the process. Because of the disconnect between β and these two germinators, this can result in events whose slots are not linked with any other in the schema.

3.7 Baseline "Germinators"

In addition to the three germinators discussed above, I added a variant of linear induction and two baselines. The variant of the LI germinator is cut off at the 800^{th} schema—this is to make it comparable to the other schemas with hard limits on the number of schemas they generate. The two baselines represent extreme, simple to generate version of schemas that aren't picky with the data they employ. One baseline generates one giant `superschema` containing all verbs in a single schema. The second baseline generates many `tiny schemas`, each being a single co-referring argument pair that was attested in the original data.

These schemas, in the process of their generation, assume that each argument slot links to another of the exact same type: each `SUBJ` is linked to each `SUBJ`, `OBJ` to `OBJ`, `PREP` to `PREP`. This heuristic was employed because the typical slot-linking process proved too slow to work appropriately with the `superschema` algorithm; it was deployed in the `tiny schemas` algorithm for consistency.

3.8 Scoring, as Implemented Here

For the most part, I follow Chambers & Jurafsky (2009) in scoring the relationships between events to be potentially induced into schemas. However, there are some noteworthy modifications, which I describe in this section.

3.8.1 Chain Typing

Following Chambers & Jurafsky (2009), I assign a single type for a particular coreference chain. As a first pass, I perform a similar procedure, selecting the most frequent head noun from the coreference chain. However, I use a slightly more sophisticated typing function, extending beyond the basic one used by Chambers & Jurafsky (2009).

This was intended to get more data out of cases that might otherwise be thrown out for lack of containing a specific most frequent head noun.

Generally, at each pass, a specific piece of information is extracted about each token—a lemma or tag of some kind—and these are accumulated as votes about what type the chain should be. If this produces a tie—both in the sense that multiple tags or lemmas have the same counts, or that all possible types and tags tied at zero—then the function falls back to the next piece of information. These are described below.

As a first pass, I gather within the coreference chain common nouns in a larger *head space* than just the head noun alone. This head space is the sequence of common nouns and adjectives between the determiner and including the head noun, but excluding anything past it to exclude relative clauses. The adjectives are excluded from consideration in typing, but are included to allow searching to continue up to the determiner. These head spaces are accumulated from all entity mentions throughout the chain, and the most frequent common noun is considered the type for the chain.

This is intended to help disambiguate cases where the head noun may not be the referent for a particular entity. For example, "police officers" and "police" may both refer to the same entity in a text—where the head noun would result in a draw, using the head space allows for a specific type to be chosen.

If one and only one type stands out as a maximum, the typing function returns that for the type of the chain.

In the event of a tie—more than one type stands out as the most frequent in the head space—the typing function falls back to the Stanford NER, performing roughly the same procedure as with the head spaces, counting up named entity tags, and if one tag type stands out as most frequent, that NER tag is returned as the type for the chain.

If the NER cannot resolve a specific entity type, the typing function falls back to pronouns. It uses a hard-coded list of pronouns and a few tokens to attempt to resolve to four different types. These are listed in Table (3.1). The same procedure is repeated with these tags. If a single tag comes out to be most frequent, then the chain is labelled with that tag.

Table 3.1: Hard-coded pronoun types. In the event a common noun or Stanford NER cannot identify the type of a coreference chain, pronouns are tagged as indicated here.

Tag	lemmas that are resolved to tag
`PERSON`	"he", "him", "she", "her", "you", "yourself", "yourselves", "yours","your", "himself","herself","themselves", "who", "whom", "Mr.", "Mrs."
`PEOPLE`	"they","them","themselves"
`SELF`	"our", "ours", "ourselves", "I","me","we","us","my","mine"
`THING`	"it", "its", "itself", "this", "that", "those"

If none of these procedures can resolve a type for the chain, a fall back `THINGY` tag is returned. While a rare event, it does end up being the most salient type on occasion.

3.9 Comparisons of Algorithm Outputs

Before evaluating the quality of the schemas described above, I want to determine how similar the schemas produced by the algorithms described above are. The goal of this section is *not* to evaluate the schemas, but rather to summarize what the different algorithms produced.

I compare these sets of schemas in three ways: first, through comparisons of similarity between sets of schemas that give a numeric measure of similarity between the

provided sets (Section 3.9.1). Next, I compare the sets to themselves and each other through a series of confusion matrices (Section 3.9.2). Last of all, given the results of these previous two sections, I will examine points of interest within and between the sets of schemas (Section 3.9.3).

3.9.1 Comparisons Between Sets of Schemas

As a first approach, one can simply see if the events contained in schemas in both are the same. Specifically, this can be done as in Formula (3.2):

$$\text{exact}(S,T) = \sum_{s \in S} \max_{t \in T} 1 \text{ iff } (s_e \cap t_e) = s_e \text{ otherwise } 0 \tag{3.2}$$

In other words, for each schema, try to find a schema in the other set that matches the first set exactly.

Table 3.2: Number of schemas shared between the output of the random walker, counter-training, and linear induction germinators.

Algorithm	Counts
RW vs LI	0
CT vs LI	0
RW vs CT	28

Out of 800 schemas each in the RW and CT batches and 14,000 in the LI batch, only 8 matched between the RW and CT schemas.

This approach has its faults. For example, a partial match—say, 5/6 events matching between two schemas—shouldn't be counted as strongly against the final result, as say a 1/6 match.

To remedy this, I propose a modification of the Jaccard index to allow for fuzzier comparisons between sets. The Jaccard index is used to compare the similarity

between two sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.3}$$

The measure should compare the extent of similarity between two sets of schemas. Comparisons between the schemas themselves can use the Jaccard index to compare the events contained therein:

$$J_e(\sigma, \tau) = \frac{|\sigma_e \cap \tau_e|}{|\sigma_e \cup \tau_e|} \tag{3.4}$$

In this case, the σ_e and τ_e being the sets of events contained in schemas σ and τ. Since σ_e and τ_e are sets, this is well-defined.

Things get more complicated at the next level, when entire sets of schemas are compared to one another. Things break down for the intersection—when deriving an intersection between two sets, partial equivalence is no way accounted for. Instead of dealing directly with the issue of partial acceptance in intersections, I redefine the cardinality of the intersection itself for fuzzy instances as:

$$|S \cap_\kappa T| = \sum_{\tau \in T} \max_{\sigma \in S} \kappa(\sigma, \tau) \tag{3.5}$$

where S and T are sets and κ is some kind of symmetric and well-defined comparison between elements of S and T. In prose, this can be thought of as "for every item in T, what is the item in S that best accounts for it?" All values of κ should be between or equal to 0 or 1. If κ is strict equivalence, then the measure reduces into the original Jaccard coefficient.

The measure is asymmetric, and as such, for a given pair of sets of schemas, must be reported as two values, where each set at one point has scope over the $\sum$. Under normal circumstances where both sets are devised under comparable conditions, these values should be relatively similar. In extreme circumstances, we should see extreme values—e.g. when one of the sets consists of a single schema while the other does not,

we'll may see one direction put the score near 1, while scoring in the other direction puts the score near 0.

For comparing sets of schemas, J_e can act as a comparison. To make this easier, I exploit the identity $|S \cup T| = |S| + |T| - |S \cap T|$. While a formal proof of this is outside the scope of this dissertation, I would like to informally discuss this to predict that it holds, even in a fuzzy case. Largely, this identity holds true because when two sets are unioned, any identical items between them are only counted once in the resulting set, so adding $|S|$ and $|T|$ alone is doubly counts items shared between the two sets. Subtracting the cardinality of the intersection accounts for this removal of duplicates. Likewise, if a fuzzy intersection is counted, subtracting the fuzzy intersection accounts for their partially shared components.

Thus, I use $|S \cup_\kappa T| = |S| + |T| - |S \cap_\kappa T|$ to define a Jaccard-esque measure J_{J_e} between two sets of schemas S and T:

$$J_{J_e}(S, T) = \frac{|S \cap_{J_e} T|}{|S| + |T| - |S \cap_{J_e} T|} \tag{3.6}$$

where S and T are sets of schemas and J_e is as defined in Formula (6.3).

With the *Fuzzy Jaccard* measure J_{J_e}, general similarities between the two sets can be more explicitly explored.

The strong affinity—though not identical matching—between the RW and CT schemas can best be explained by their selection criteria.

Table 3.3: Fuzzy Jaccard values between sets of schemas generated using different algorithms on the NYT corpus.

S vs T	$J_{J_e}(S, T)$	$J_{J_e}(T, S)$
RW vs CT	0.287	0.215
CT vs LI-Trunc	0.0535	0.0492
RW vs LI-Trunc	0.0963	0.0540

3.9.2 Confusion Matrices

While the numbers reported above are helpful, what do they mean, exactly? The Jaccard-esque measure used is novel, so its interpretation is unclear. To understand more deeply what's going on, this section presents a few large confusion matrices comparing the content of the outputs of each of the schema germinators presented. These illustrate both the internal and external similarities of their outputs.

Figures (3.2 – 3.4) contain self-similarity matrices for each of the germinators presented—that is, a comparison between each of the individual schemas generated by each algorithm and their similarity to one another. The schemas themselves are sorted alphabetically, by the order of their events sorted alphabetically. Each pixel filled on a grayscale based on the number of shared events between the two schemas assigned to each row and column; these values are what a searched through to determine the Fuzzy Jaccard value. In the self-similarity comparisons, along the diagonal, each schema is compared to itself, resulting in a bright white line appearing along the diagonal. These may appear to vary in value, but these changes are illusory—close inspection will reveal that apparent changes in the diagonal of the self-similarity confusion matrices are actually due to lighter and darker values appearing in the context of the diagonal.

The linear induction (LI) germinator produces schemas with minimal overlap between one another, resulting in a nearly one pixel-wide line down the diagonal—that is, other than themselves, the schemas don't share much in common. Note that the list presented in the confusion matrix is truncated, as a large, long tail of single-event schemas is produced by the LI germinator, and the LI germinator generates thousands instead of the fixed 800 generated by the CT and RW germinators. The truncated schemas are much easier to visualize.

Schemas generated with counter-training have some scattered overlap—a few blocs of highly similar schemas that sorted together as squares along the diagonal. The sorting heuristic applied here caused some schemas that should have belonged in a bloc to be placed elsewhere, resulting in one pixel wide "echoes" of tight blocs scattering into the void beyond the diagonal.

Random walker generated schemas, on the other hand, have strong but variable overlap. Instead of presenting large, bright blocs, the RW-schemas produce irregular, plaid-like blocs of variable similarity. This is, of course, a product of the technique—more probable verbs bubble up to join perhaps rare seed words, and they bring along in chains more general seed words. The end result is more similar schemas. However, a few schemas do exist that are themselves unique, cutting black lines throughout these blocks of homogeneity. Some of these are tight, near-white blocs like those found in the counter-training matrix. The random walker produced many schemas, but replicated the statistical distribution of those schemas more closely than its competing germinators.

Of course, the outputs of each germinator can be compared to itself, thereby showing similarity throughout the set of schemas as a whole. In other words, such a self-comparison can show whether a particular germinator is making unique schemas or generating very similar schemas again and again. In Figure (3.5), there are some similarities between the massive self-similarity bloc of the RW-schemas and a handful of the counter-trained schemas. A few schemas seem to be similar along the shifted "diagonal" of both algorithms. The two comparisons between the linear induction germinator and the other two algorithms (Figures 3.6 and 3.7) are generally un-interesting—look closely, and a peppering of dim gray pixels may be visible—indicating the the sets of schemas did not share much in common with linear

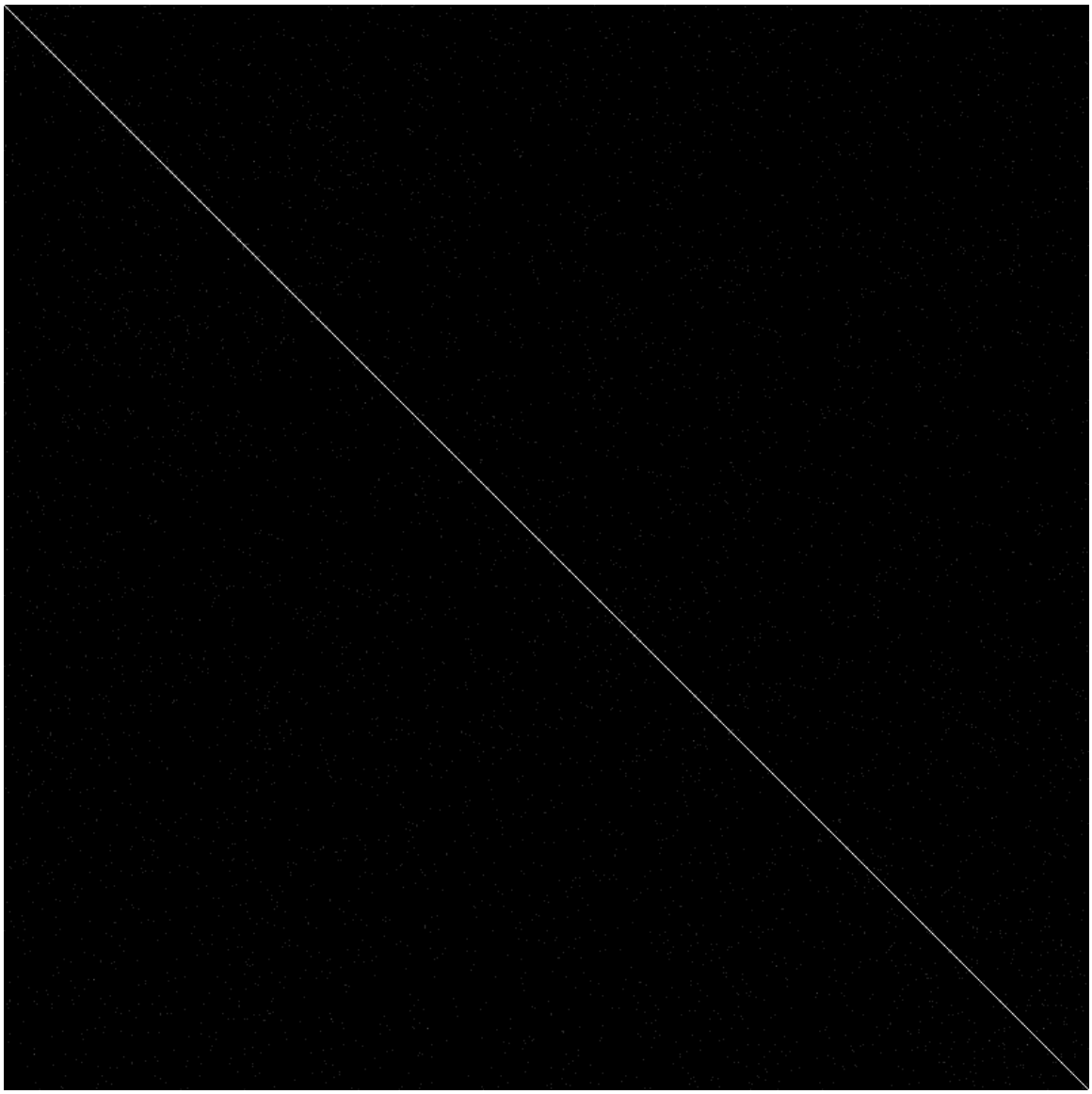


Figure 3.2: Linear induction (truncated) self-similarity matrix.

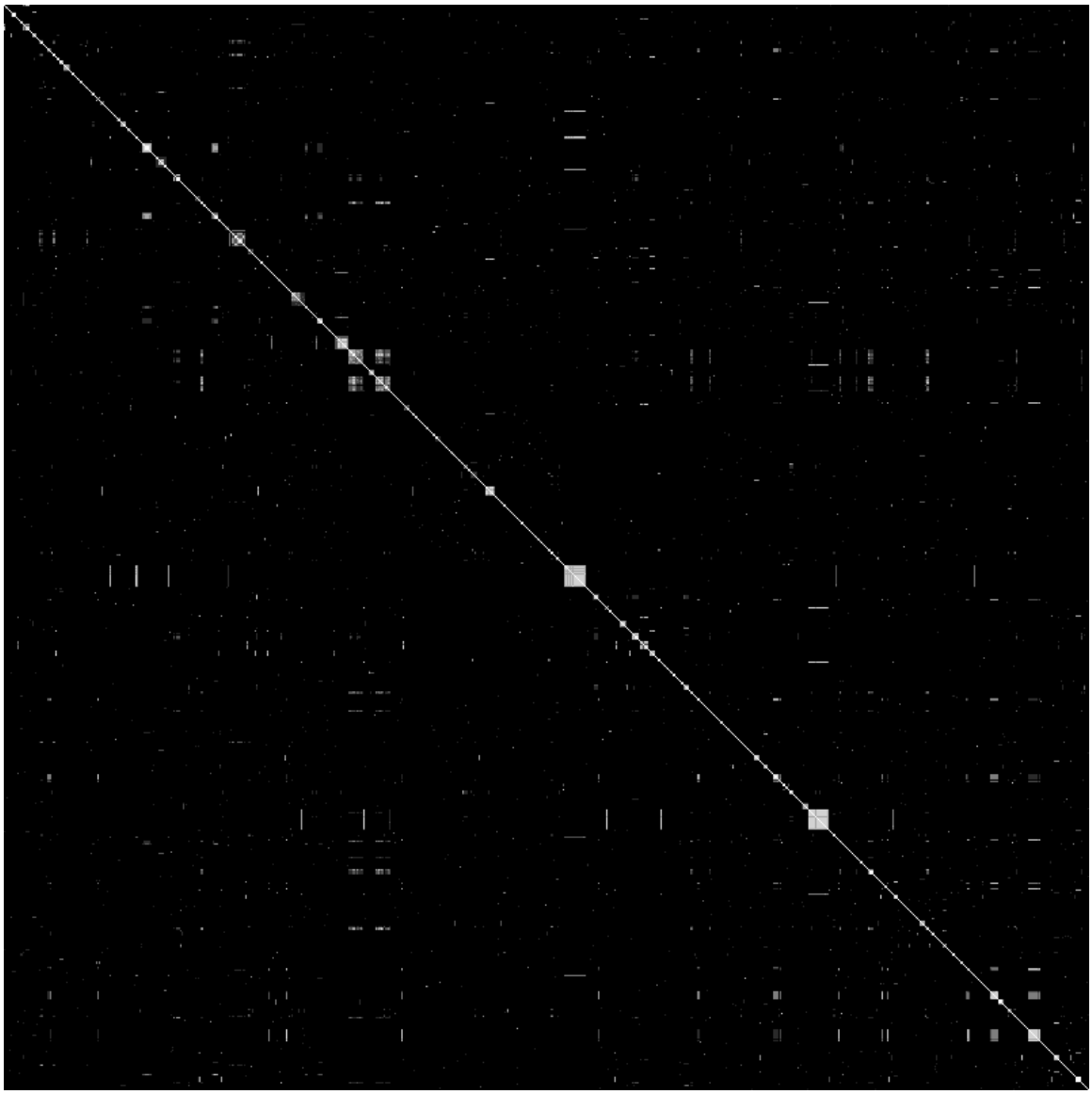


Figure 3.3: Counter-training self-similarity matrix.

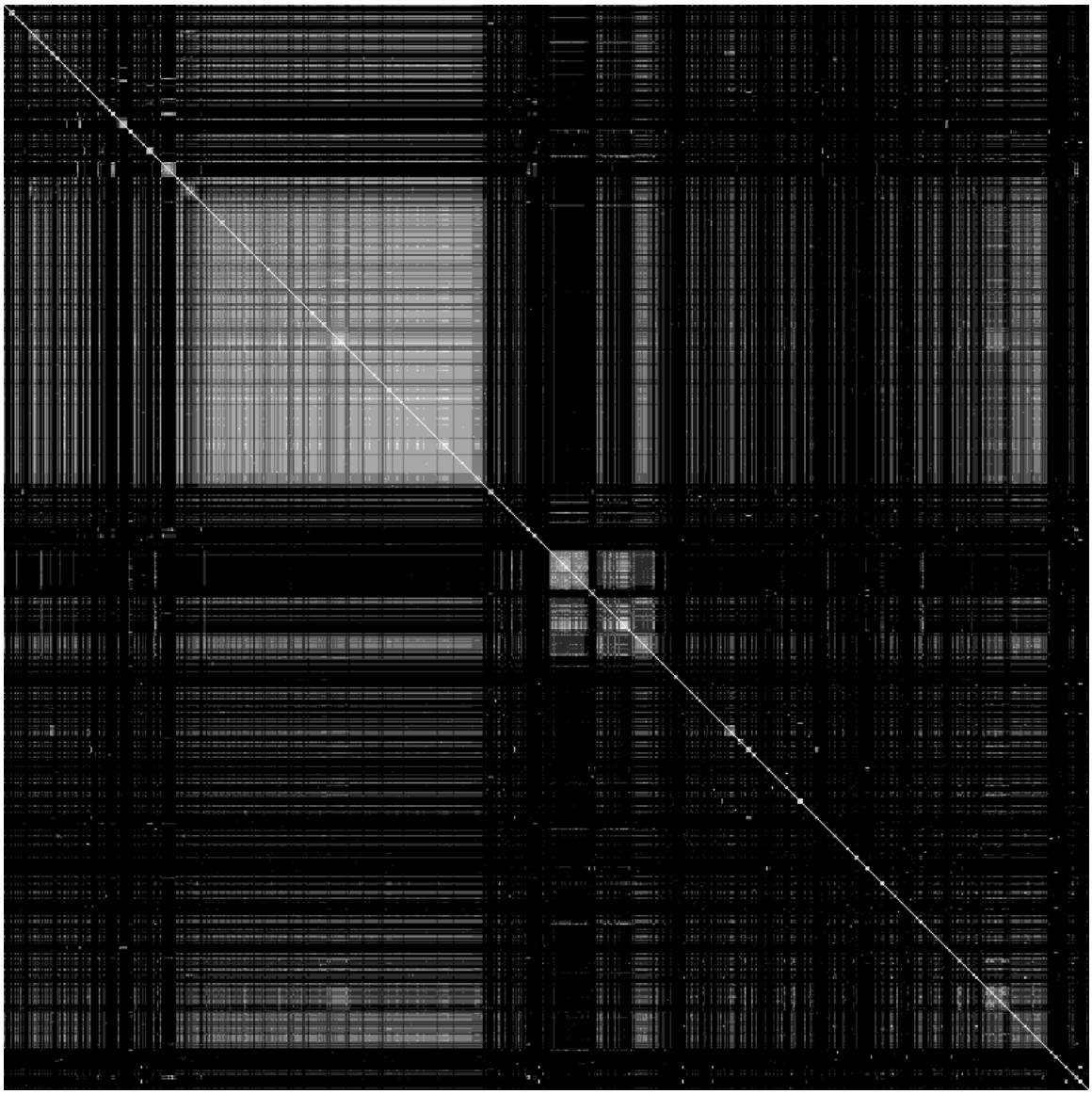


Figure 3.4: Random walker self-similarity matrix.

induction schemas. This indicates that the counter-trained and random walk schemas share much more in common than they do with linear induction schemas.

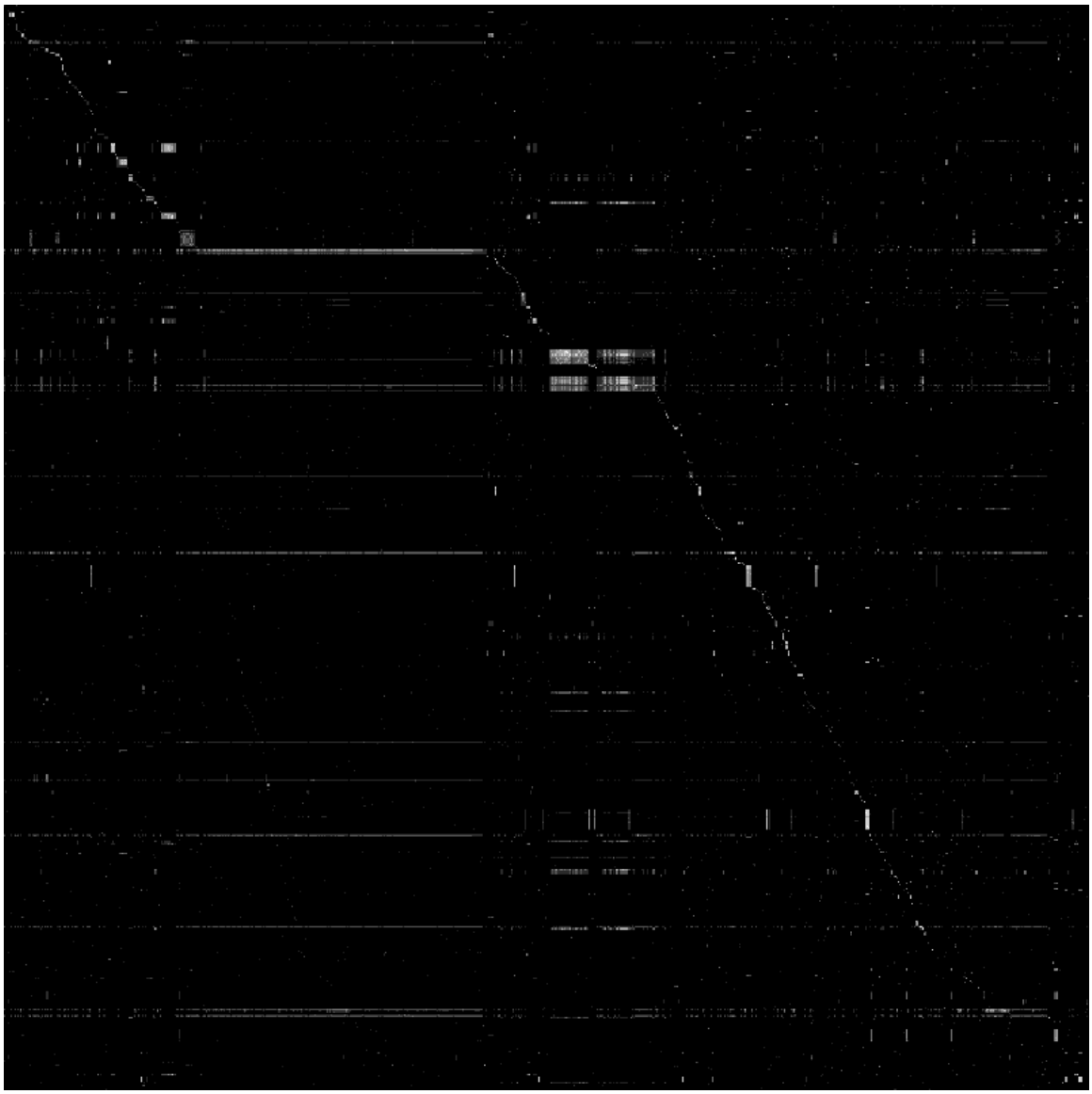


Figure 3.5: Confusion matrix between counter-training and random walker.

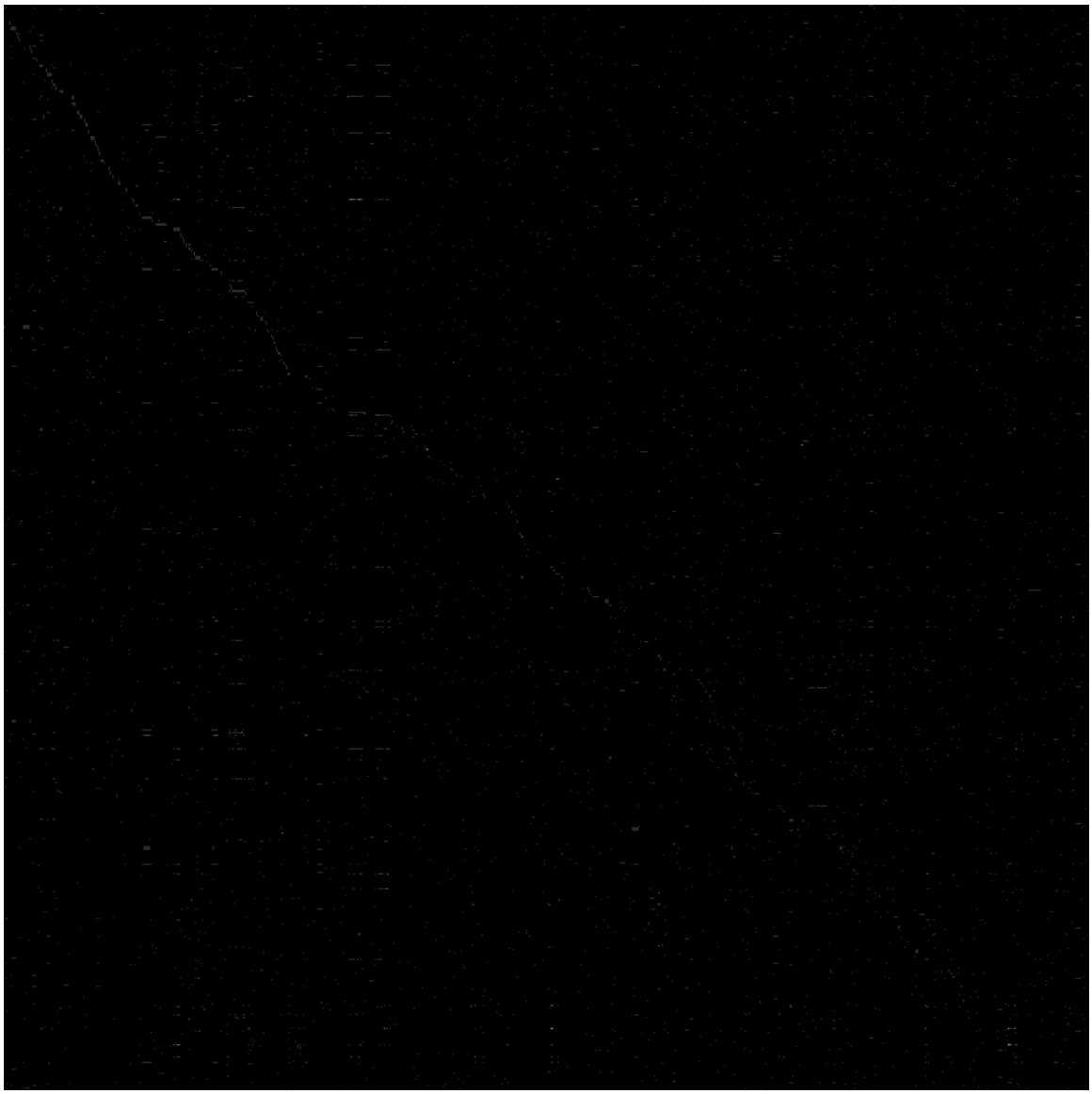


Figure 3.6: Confusion matrix between linear induction truncated and counter-training.

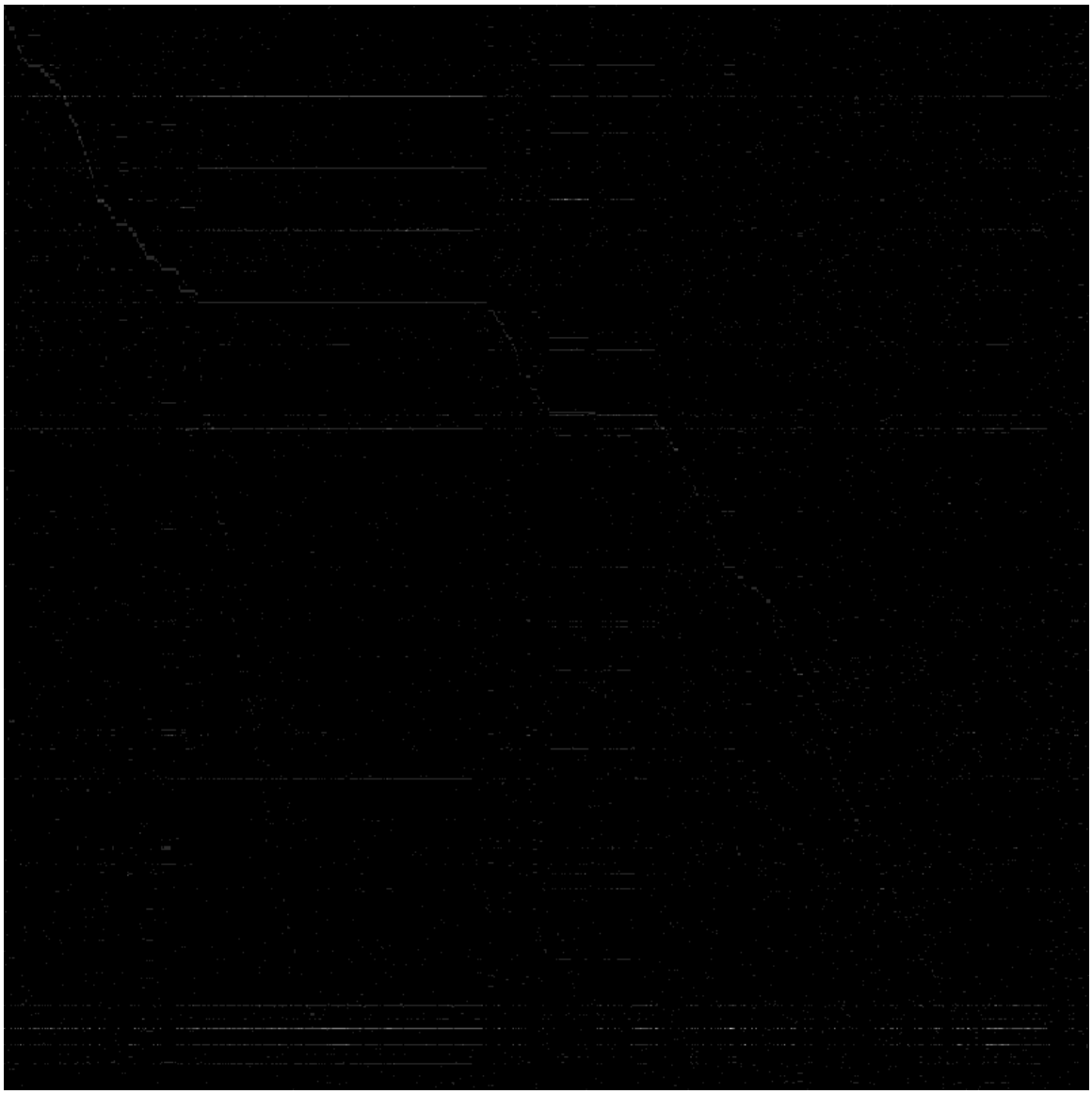


Figure 3.7: Confusion matrix between iinear induction truncated and random walker.

3.9.3 Examples of Schemas at Points of Interest

CT Bright Blocs

The counter-training algorithm tends to generate two types of "structures" when its output is illustrated with a confusion matrix. The first are tight, bright blocs around the diagonal. Sometimes, the heuristic used to generate the confusion matrices don't capture these blocs well, resulting in bright similarity lines at a distance, away from the diagonal. Either way, these indicate a strong cluster of very similar schemas.

Some schemas contained in these bright blocs are illustrated in Figure (3.8). The meaning and source of some of the generated schemas are pretty clear: for example, the schemas contained in Figure (3.8, Row 1) clearly reflect democratic processes in one form or another—the procedures surrounding negotiation and voting:

> " Those critics blame the Pride Agenda , the statewide lobbying group that has made *passing* the bill its top priority , for what they see as an inexcusable omission . By failing to work to *amend* the bill... The Assembly has repeatedly *approved* the bill in its current form... he would *vote* for the bill regardless..." — The New York Times, 2002-12-16[1]

> " mechanics *voted* 57 percent to 43 percent to reject cuts... the concession package would *pass* the second time around... In September, mechanics at US Airways *approved* their concessions to the airline..." — The New York Times, 2002-12-02[2]

[1] "On Eve of Vote, Gay Rights Bill Is Besieged From Within," document id: 1449269 Sandhaus (2008a)

[2] "United Meets Leaders of a Holdout Union," document id: 1445332 Sandhaus (2008a)

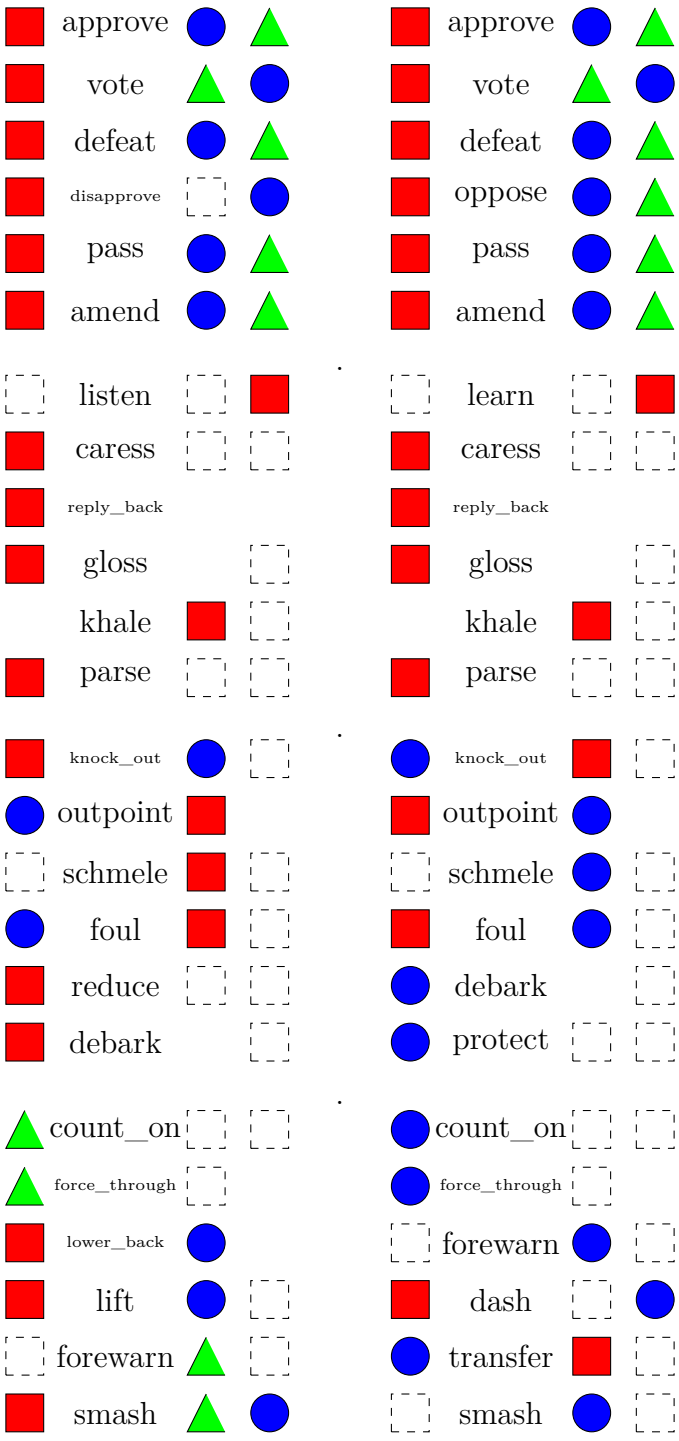


Figure 3.8: Schemas from "bright blocs" generated by the counter-training technique.

But not all schemas are straightforward—some are more subtle than others. Some schemas might contain "errors" in the sense that the connections drawn by the germinators are unintuitive or undesirable, but these errors are often systematic reflections of the discourse embodied within the schema itself. Notably, the schemas tend to contain more obscure "verbs," such as "khale" and "schmele," likely mislemmatizations of "Khaled" and "Schmeling." For (3.8, row 3), we're seeing words related to—or tangentially related to—sports, including "Schmeling," who was a boxer:

> "Sharkey landed a low blow in the fourth round, *knocking Schmeling out* , but giving the German the title by *foul* ." — The New York Times, 1994-08-19[3]

> "By slacking off on defense, he can *reduce* the spread, the number of points that bookmakers have predicted... 'We can't *protect* ourselves against the kid who thinks he's above the law'... 'No wonder it scares them every time one of their players seems to lose concentration, misplaces his man, *fouls* unnecessarily.' " — The New York Times, 2002-12-22[4]

> "Make it two *fouls* on the third strike, and you 're out . This will offset the advantage to the hitter of the three-ball rule . Reduce the time that is wasted by batters ' *protecting* the plate. ' " — The New York Times, 1994-09-17[5]

Though this schema doesn't strictly encompass sports:

> " 'We are systematically *fouling* our nests,' said Joanna D. Underwood... 'It was not until the 1970 's that they were linked to deterioration of the

[3]"Jack Sharkey, Boxing Champion, Dies at 91," document id: 707113 Sandhaus (2008a)

[4]"BackTalk; Conditions Favorable For Point Shaving," document id: 1450895 Sandhaus (2008a)

[5]"'When Baseball Returns, Speed Up the Game," document id: 712584 Sandhaus (2008a)

ozone layer that *protects* the earth from solar radiation..." — The New York Times, 1995-03-15[6]

"They should have said that our city, like the whole United States, is *fouled* by guns... stinks of them. Say it plain: Until we do something about that deathly stench the men and women we hire to *protect* us will be doing their duty knowing that every minute of every tour they are surrounded by muggers, killers and thieves in greater number, expertise and firepower than the world has ever experienced." — The New York Times, 1994-08-26[7]

" 'the workers are just trying to *protect* their interests,' said Victor Cienfuegos, a Managua photographer in his early 30's, who said he was a strong Sandinista supporter. 'There may be an element of the Government using this to *foul* up the new government' — The New York Times, 1990-04-24[8]

In this sense, the notion of "foul" floats between the sports notion of committing a violation of the rules and the notion of polluting—literally or figuratively.

In Figure (3.8, row 2), we see a mixture of words that seem strange on the surface. However, critics—quite possibly the same critics—of performance arts use verbs like "caress," "gloss," "parse," and "learn" together often:

[6]"A Study Calls Household Materials Especially Toxic," document id: 748828 Sandhaus (2008a)

[7]"On My Mind; Why Two Cops Were Shot in the Subway," document id: 708498 Sandhaus (2008a)

[8]"Strikes Strangling Nicaragua And New Leaders Cry Foul," document id: 346517 Sandhaus (2008a)

“that *caress* by Lifar is repeated in a clever *gloss* on Act I of ” Giselle , ” Mr. Eifman ’s conception coalesces with dramatic force... One might suppress a laugh when Lifar ’s Albrecht here *caresses* Wilfrid, his squire.”

“ When he later must overcome some formidable obstacles to win his bride ’s love , Mr. Malas uses his voice as a *caress*, playing down its power in favor of its tenderness... ’ Much of the Broadway showmanship of Mr. Gutierrez ’s “ Fella ” has been brought to a higher *gloss*...”

“Gradually , however , as he *learns* the business , his attention turns to Jean-Marie, and the film takes on a darker mood as the two men try to *parse* currents of friendship, mentorship and passion... As far from the *gloss* of Hollywood as they are...”

“In Great Britain , meanwhile , both black and white performers *learn* soul from American records as a pop style... Roachford , the group built around the singer-songwriter Andrew Roachford , reclaims 60 ’s and 70 ’s soul with the*gloss* of 80 ’s synthesizers...”

“This morning , my daughter had an attack of hysteria... In horror and despair, my wife and I tried to hold and *caress* her...It would be a year later before I would *learn* about my daughter, Thuy Duong, and two years after that before I would return to Vietnam on a Christmas visit .”

“you can *learn* anything from a book – or nothing . You can *learn* to be a suicide bomber , a religious fanatic or, indeed, a Bush supporter as easily as you can *learn* to be tolerant, peace-loving and wise... You can *learn* to be a sexist or a feminist... We do better to argue with them than to *caress* their spines.”

> "Across the way is a suite of especially beautiful images devoted to an attractive young woman we *learn* is Siobhan , the artist 's next romantic involvement... ''For me taking a picture is a way of touching somebody—it's a *caress*.'

"khale" is likely a mislemmatization of "DJ Khaled," appearing strongly with the critics. All-in-all, it seems the schemas themselves are collections of verbs that are often used in critical reviews of artistic works, reflecting a sort of euphuistic language to describe the process of enjoying such work.

As for Figure(3.8, row 4), there is a pair of schemas that contain many words involving some sort of Schankian `PTRANS`, but how are they related? When is something "transferred," "dashed," "forced through," and "smashed?"

> " The changing technology has led companies like Human Relations Media , Guidance Associates of Mount Kisco and Sunburst Communications , also of Pleasantville -LRB- headed by Mr. Schloat 's brother , Warren -RRB- to *transfer* many of their films to videotape... But those who dream of fame and fortune starring in school videos should be *forewarned* ." – The New York Times, 1988-01-24[9]

> "Drain excess liquid from the shrimp and grouper and *transfer* to a large bowl... *transfer* to paper towels to drain... 10 cloves garlic , peeled and *smashed* " — The New York Times, 2000-09-17[10]

> "The son of Jamaican immigrants , Andre Fletcher attended the Bronx High School of Science , but later *transferred* to Brooklyn Technical High

[9] "SCHOOL VIDEOS OPEN NEW DOORS FOR PRODUCERS AND ACTORS," document id: 112424 Sandhaus (2008a)

[10] "Food; Kitchen Confidential," document id: 1230967 Sandhaus (2008a) (and for the record, this fell under the `onlineproducer` category *Labor*).

School... He also talks of dreams, now *dashed* , that the two brothers had of modeling..." — The New York Times, 2001-12-26[11]

It seems that overall, the schemas indicate a number of words that are used largely metaphorically, except for the odd recipe that's classified as a `Labor` article.

RW Superbloc

One point of interest is the massive bloc visible in Figure (3.4). These schemas typically share anywhere between 3–4 events in common. Four of them have been pulled out for examination in Figure (3.9). Each of the four schemas contain the verb `ask`; given the orthographic arrangement heuristic used for creating the confusion matrices, this is likely the "cause" for the bloc. They also all contain `give`—3/4 contain `meet` or `take`. The massive bloc and the "bright bands" extending off of it reflect the distribution of these light verbs throughout. Given the size of this bloc, nearly 427 of the 800 schemas in the random walker schema set contain at least one of these verbs. This suggests potentially treating the verbs contained in this bloc and their objects as multi-word expressions. However, deciding what is and is not a light verb is a non-trivial problem and outside the scope of this dissertation.

A clear question ask here is why these schemas are so generic, especially compared to their counter-trained counterparts. The clearest reason is the math behind the probability distribution used in the random walker. To make the schemas more probabilistic, the *pmi* weights are converted from log to more linear values by exponentiation (Formula 3.1), which are then converted to probabilities. In a log space, the weights behave close to a uniform distribution of events; when sampled, the events are drawn out qualitatively seem to bear little connection those in the rest of the schema.

[11] "A Handy Bond Trader, a Kitchen Whiz Technician, a Ponytailed Doctor," document id: 1354361 Sandhaus (2008a)

However, the linear space inflates a subset of events that are likely to appear with many different things, and the random walker lacks any internal forces to push back against such generic schemas. Drawing more precise distinctions between the verbs that appeared in this bloc may prove the best option for breaking it up if desired.

These sorts of generic schemas are not necessarily harmful or inaccurate. It is reasonable that, with any verb, there is a good chance these events will appear along with them. The meaning of these words, however, changes a lot within context, as they often have a metaphorical usage.

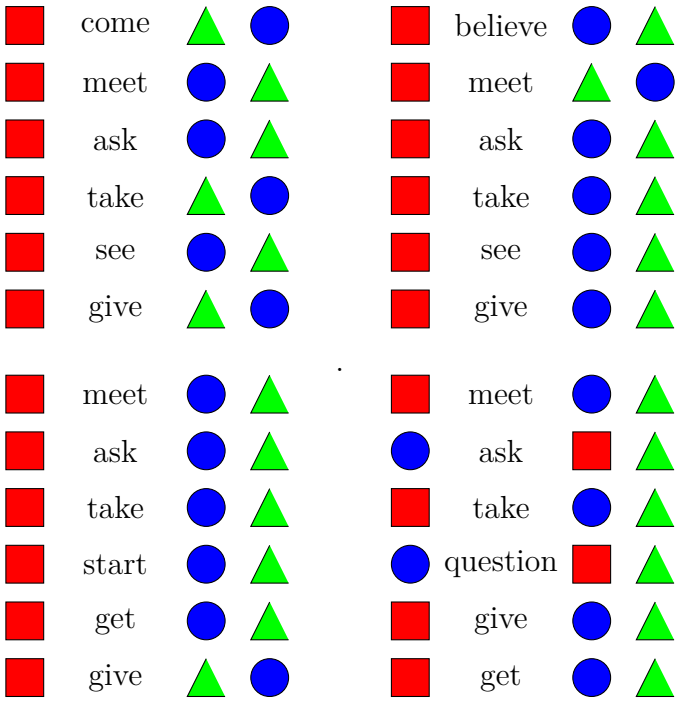


Figure 3.9: A sample of a few of the schemas from the large bloc in the RW set. The large bloc seems to generally reflect the shared verbs `give`, `take`, `meet`, and `ask`.

These are often used in text, many in the same articles.

> “ I keep *asking* when someone stands and shoots someone point blank how could he not be found not guilty of attempted murder and assault ,

"... " There is a lot of controversy going on here , and we don't want to *see* anyone not *get* due process of the law like Father Blackwell, but we are also trying to be fair with this , *given* Dontee 's case . "... " *took* an action that a lot of people around the country in this sex abuse scandal wanted to *take*, and that does not make him a hero , but understandable . " — The New York Times, 2002-12-18[12]

" It was because of what happened 11 years ago , *getting* arrested . " I think a lot of G.M. 's were scared away . I think the ones that have stepped up and helped me with my career – Bobby Clarke , Craig Patrick and especially now Glen Sather , who 's *given* me another shot... '*Give* him a chance... I've *seen* a lot of people crazier than him that I 've handled ." — The New York Times, 2002-12-18[13]

"The security officers who *took* part in the survey work in financial , retail , health care and many other fields... The survey questions do not *give* a sense of what information might be shared or under what circumstances... growing concerns about government encroachments on privacy and civil liberties have not *taken* into account the degree to which people hand over information willingly , said Mark Rasch , a former federal prosecutor... " We 've been so worried about *giving* them extra power and authority without worrying about what they can do with no extra power and no extra authority , just by *asking* " – The New York Times, 2002-12-18[14]

[12]'Acquittal in Shooting Of Priest Splits a City' document id: 1449683 Sandhaus (2008a)
[13]'Rangers Give Troubled Player a Shot' document id: 1449736,Sandhaus (2008a)
[14]'Some Companies Will Release Customer Records on Request' document id: 1449741, Sandhaus (2008a)

While the verbs themselves seem to indicate a Schankian `ATRANS` or things that facilitate such actions (e.g. "ask") they are light verbs, as shown in the examples. Such persistent metaphor is not likely accidental (Lakoff & Johnson, 1980), but in-depth discussion of the various metaphors on which language is built is beyond the scope of this dissertation.

3.9.4 Duplicate Schemas

The results presented in the prior sections make evident a number of duplicate schemas. One possible variant of the counter-training germinator and the random walker germinator is one with a post-processing step that merges schemas that contain a substantial number of similar events. While this on the surface seems a simple and natural step, there are a lot of considerations that make it less trivial than it appears.

Early in the process of writing this dissertation, the duplicate schemas were found subjectively helpful in understanding the content generated through germination. A schema appearing multiple times had a way of highlighting the content contained in that schema, showing that the schema itself was not a random variation but a consistent feature derived from the data.

In considering merging identical schemas, this made the idea of schema merging at the least requiring some sort of weighting scheme to retain this feature of the results. Given that the model used here quite literally followed the Chambers & Jurafsky (2009) definition of a schema—a tuple between a set of events and a set of chains—it would require a great deal of refactoring to support a definition of a schema that was more generally capable of containing more information since many components of the code underlying this dissertation depend directly on unpacking the 2-tuple structure of each schema.

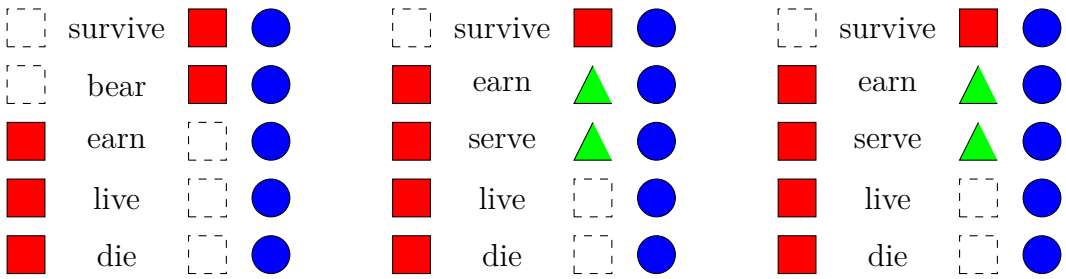


Figure 3.10: Three schemas generated via counter-training. These were taken from the peak performer set in the hill-climb experiment in Section (4.7).

In the event of schemas that are not exactly alike, but are nearly alike, the possibility was considered of more structurally informed mergers. For example, consider these three schemas in Figure (3.10). These schemas are derived from the Obituaries section of the New York Times. "survive" is from the idiom "X was survived by Y," and "live" and "die" are clear topics of discussion in an obituary. The schemas differ in their containing "bear"—a lemmatized form of "born"—and "serve." Two of the schemas are identical—the one on the left is different. While a merger wouldn't necessarily invalidate the content of the right two schemas, it is not necessarily the case that the schema on the left has the same entailments. In other words, everyone is born, but not everyone who is born necessarily serves. In merging the schemas without considering divergent timelines and possible worlds, the potential to make this distinction in our model is lost.

Given the open-ended nature of such questions, I considered addressing these issues in later chapters. However, as the dissertation progressed, the focus narrowed more specifically to the interaction between document categories, topics, and schemas. The duplicate and nearly duplicate schemas were retained as a property to be studied

rather than an error to be reconciled. Perhaps in future work, this problem can be directly addressed in the nuanced manner required.

3.10 Conclusions

In this chapter, I introduced the distinction between score and germinators in creating narrative schemas. A score ranks the similarity between a candidate event with a schema; a germinator interprets the score and determines how candidate events are traversed, turning the mere relation between events and schemas into fully generated schemas. I introduce three germinators—techniques for using a relationship between an existing schema and a candidate augmentation—and show that differences in such techniques can affect the content and quality of the output schemas in dramatically different ways.

Three germinators were discussed: the linear induction germinator, the counter-training germinator, and a random walker germinator. The linear induction germinator descends the list of verbs from most to least frequent, adding each event to a small set of growing schemas. The counter-training germinator generates an entire set of schemas simultaneously, penalizing event inductions that are more generally felicitous, but providing the opportunity for more general additions. The random walker germinator converts the score between a schema and its candidate events into a quasi-probability and chooses events to add at random. Both the counter-training and random walker germinator consider all possible event augmentations for each schema simultaneously, while linear induction only considers a handful at once. Two baseline germinators were considered: a superschema—which lumps all events in one giant schema—and tiny schemas, which puts each attested in co-referring argument pair into a separate schema.

I showed that, while there are some similarities between the schemas generated, the three germinators produce schemas that are fundamentally different both quantitatively—by using a modified form of the Jaccard coefficient that assigns credit for partial overlap between set elements—and qualitatively—by examining the different sorts of schemas generated by the germinators and the contexts from which they are likely derived.

In the next chapter, I examine these schemas further using two evaluations. These use data that was held out before these schemas were generated.

Chapter 4

Evaluating Schemas

4.1 Introduction

Thus far, little work has been done evaluating schemas directly. The cloze task was devised as a metric for evaluating the narrative language model (Chambers & Jurafsky, 2008, 2009), but the output schemas themselves were never directly evaluated. Only Balasubramanian et al. (2013) evaluated schemas via Amazon Mechanical Turk arguing that "there are no good automated ways to make such judgements." The experiments undertaken were the first to validate that non-expert human beings recognized schemas as reflecting something real.

However, such manual evaluation is slow and expensive. Since I set out in this dissertation to use schemas themselves as interpretable, interchangeable components in the analysis of the narratives of corpora, reliable automatic techniques for evaluation of schemas are essential to quickly and reliably measure improvements to the techniques that generate them. This ensures the best, unsupervised results down the road and is a proxy for the quality of analyses done with the schemas evaluated.

In the Chapter (3), I defined schema germination and showed the choice of germinator can yield vastly different schemas. Previously, script models were evaluated with the cloze task. However, despite different germinators producing different schemas, the cloze task would give the exact same result for all of those schemas generated since

it evaluates the underlying score used to measure the similarity between candidate events and schemas.

In this chapter, I attempt to evaluate those schemas. I do this with two different techniques; the *narrative argument salience through entities annotated* (NASTEA) task (Section 4.3) and a pair of information-theoretic *minimum description length* (MDL) measures (Section 4.4).

With respect to both evaluations, I dig deeper than simply using them to evaluate schemas on the NYT corpus (Sandhaus, 2008a). For NASTEA, I attempt to push the boundaries of performance with a parameter climb experiment, detailed in Section (4.7), and I provide some simple baselines to better contextualize NASTEA scores in Section (4.6). For MDL, I use documents annotated with gold standard data from the OntoNotes corpus as well as those same documents with automatic annotations to see if given better data, the MDL measures will reflect the improvements (Section 4.8).

Both of these rely on identifying the *presence* of a schema in a document which—unlike identifying the presence of a token of a given type in a document—is a non-trivial problem to address. I define what I mean by presence in Section (4.2), which is where I begin in the next section.

4.2 The Presence of a Schema in a Document

One of the goals of this chapter is to define new evaluations of schema quality. I define two in this chapter, both of which rely on holdout documents to conduct the evaluations. In the NASTEA task (Section 4.3), schemas are used to extract entities from documents, extractions which should agree with human annotations of salient

entities. In the minimum description length measure (Section 4.4), schemas are used as an encoding to capture the events described in a document.

In both cases, some sort of measure of *presence* is needed to determine what schemas should be applied to which documents. This presence measure should be modular enough to apply in both tasks.

Determining whether or not a word or n-gram appears in a document is a relatively simple task, but identifying whether a narrative schema is present or not is neither trivial nor categorical. The notion of presence is essential to the NASTEA task where it identifies which schema to use to identify the salient entities. It is also necessary to select which schemas are relevant in encoding a document's events in the MDL measure.

In the following sections, I deploy a measure of *presence* that reflects the *canonicality* of a document—that is, how closely a document matches a schema. This measure uses the events of a schema as a proxy for its content—excluding the arguments from the measure. I explicitly exclude coreference information from the measure since coreference is error prone; while I trust it *en masse* for generalizing over many documents, I am not so sure coreference can be trusted on a document-to-document basis.

Measuring the presence $p_{S,D}$ of a schema S in a document D begins with $V_{S,D}$, the set of verbs in D that represent events in S:

$$V_{S,D} = \{v_i : v_i \in D \wedge v_i \in S_e\} \tag{4.1}$$

where $v_i \in D$ is true when an instance of verb v_i is inside document D. S_e is the set of events in a schema, each represented by a verb. The same verb type can appear multiple times in the set, as each instance is uniquely indexed. A sentence can have multiple verbs, and all relevant verbs are included in $V_{S,D}$.

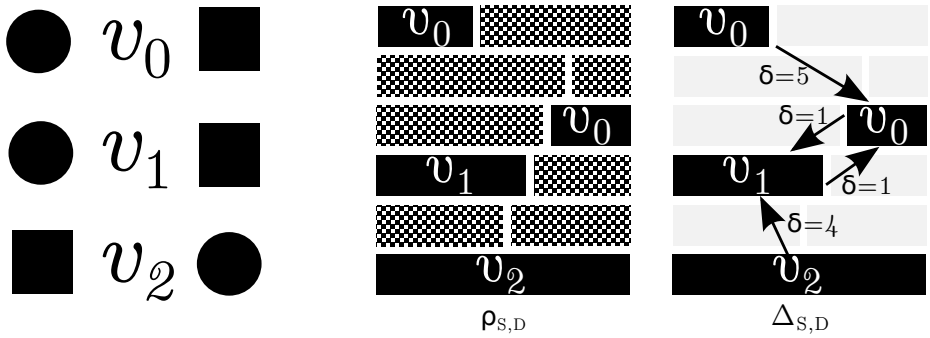


Figure 4.1: An illustration of how a document looks through the two components of schema presence. In other words, it is how the document D looks through density $\rho_{S,D}$ and dispersion $\Delta_{S,D}$ for a hypothetical schema S. In D, a rectangular block represents each sentence. v_i in that each rectangular block indicates an instance of the verb corresponding the v_i event in S. The checkerboarded sentences in ρ contribute to the calculation through $|D|$; these sentences are grayed out in Δ to indicate that they only indirectly participate in the calculation by increasing the distance for various δ values.

There are two ways to consider the distribution of verbs within a document, both of which contribute to defining presence: *density* and *dispersion*. Density ρ is defined as:

$$\rho_{S,D} = \frac{|V_{S,D}|}{|D|} \tag{4.2}$$

In other words, $\rho_{S,D}$ measures how much of the document D is composed of verbs $V_{S,D}$ representing the events in schema S. If this factor is high, then the document as a whole is very close to being only the series of events expressed in relevant schema.

While a high density value is a strong indicator of presence, some cases where the density is not as high may still be interesting. If a set of relevant verbs are close together, this indicates some expression of the schema, while a disperse set of verbs are less likely to be an expression of the events listed in the schema. This I call dispersion

Δ, defined as:

$$\Delta_{S,D} = \frac{1}{|V_{S,D}|} \sum_{v_i \in V_{S,D}} \min_{v_j \in V_{S,D} - \{v_i\}} \delta(v_i, v_j) \tag{4.3}$$

where $\delta(v_i, v_j)$ indicates the distance in sentences between two verbs v_i and v_j. The minimization seeks to find the nearest v_j to v_i in $V_{S,D}$, which is computed for every v_i contained in $V_{S,D}$.

The presence measure should be higher for those documents in which the elements of a schema are both dense (throughout the document) and not disperse, I define *canonical presence* p as:

$$p_{S,D} = \frac{\rho_{S,D}}{\Delta_{S,D}} \tag{4.4}$$

This defines the extent to which a schema is present in a document—more specifically, the degree to which a document itself comes close to being an exemplar of the schema. The components of p are illustrated in Figure (4.1).

4.3 Evaluation: Narrative Argument Salience Through Entities Annotated (NASTEA)

As Balasubramanian et al. (2013) point out, there is no "good" existing way to evaluate schemas. The cloze task is a primarily a diagnostic metric, not an evaluation (Chambers, 2011). Most prior work, as discussed in Chapter (2), focuses on improving performance on this task or variants of this task and does not address schemas themselves.

In this section, I propose a task that is solvable, evaluates schemas directly, and concerns an aspect of narrative orthogonal to what the cloze task involves—the participants. Salient entity annotations in the New York Times corpus, performed by

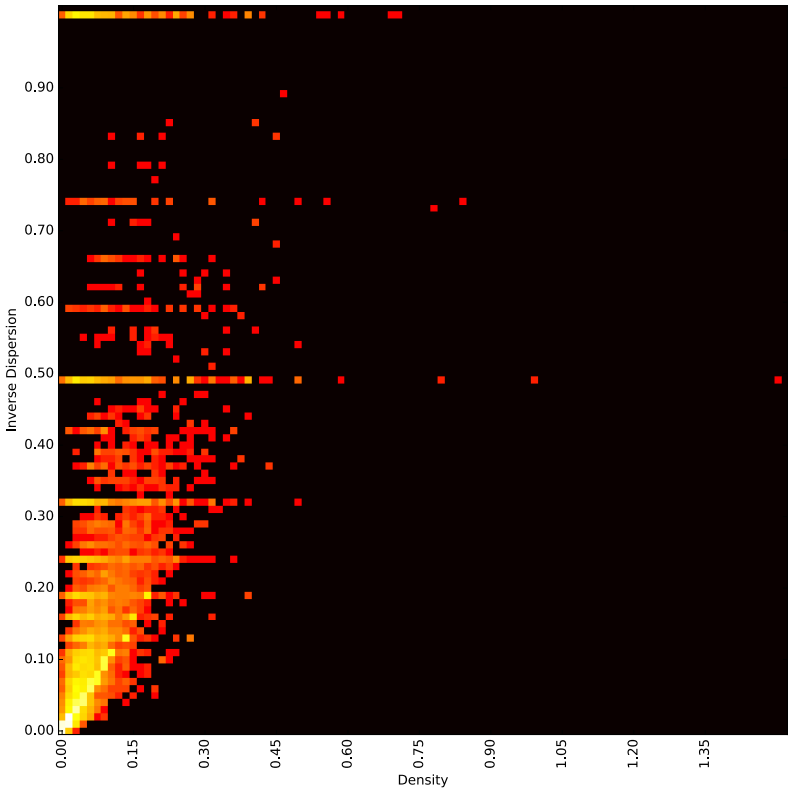


Figure 4.2: A heat map of presence values for all schemas in the NYT Corpus using all three germinators. Color determined by the natural logarithm of the counts of components of presence values that occurred at each value.

library scientists,[1] appear well-suited to this task. I investigate whether narrative schemas identify these salient entities, under the assumption that entities deemed important by the annotators indicate *Narrative Argument Salience Through Entities Annotated*, or *NASTEA*.

Since I addressed the issue of presence previously (Section 4.2), devising the NASTEA task boils down to two issues: how to extract entities from a document using a schema (Section 4.3.1), and interpreting the results of multiple extractions (Section 4.3.2).

4.3.1 Extracting Salient Entities with a Schema

The whole process for using schemas to extract salient entities is illustrated in Figure (4.3).

Once schemas have been ranked for presence, the best match must be applied to the matching document in some way. I use the verb/dependency pairs found in that document that are also present in a schema to extract entities of importance. From each pair, any NP governed through the indicated dependency is extracted in whole. Only NPs containing proper nouns (`/NNP.*/`) are retained, as common nouns are not indicated in the NYT Metadata.

One side effect of linear induction germination is a large number of schemas containing only a single verb—having only weak connections with the events in any other schema. I exclude these schemas from the NASTEA task.

The entities extracted are compared with the entities indicated in the NYT Metadata, a union of the `person`, `organization`, and `location` tags for each document. Each person, organization, or location from the metadata is tokenized with NLTK's

[1] The NYT Corpus documentation gives little description of the salient entity annotation process, except that "[t]hese tags are hand-assigned by The New York Times Indexing Service" (Sandhaus, 2008b).

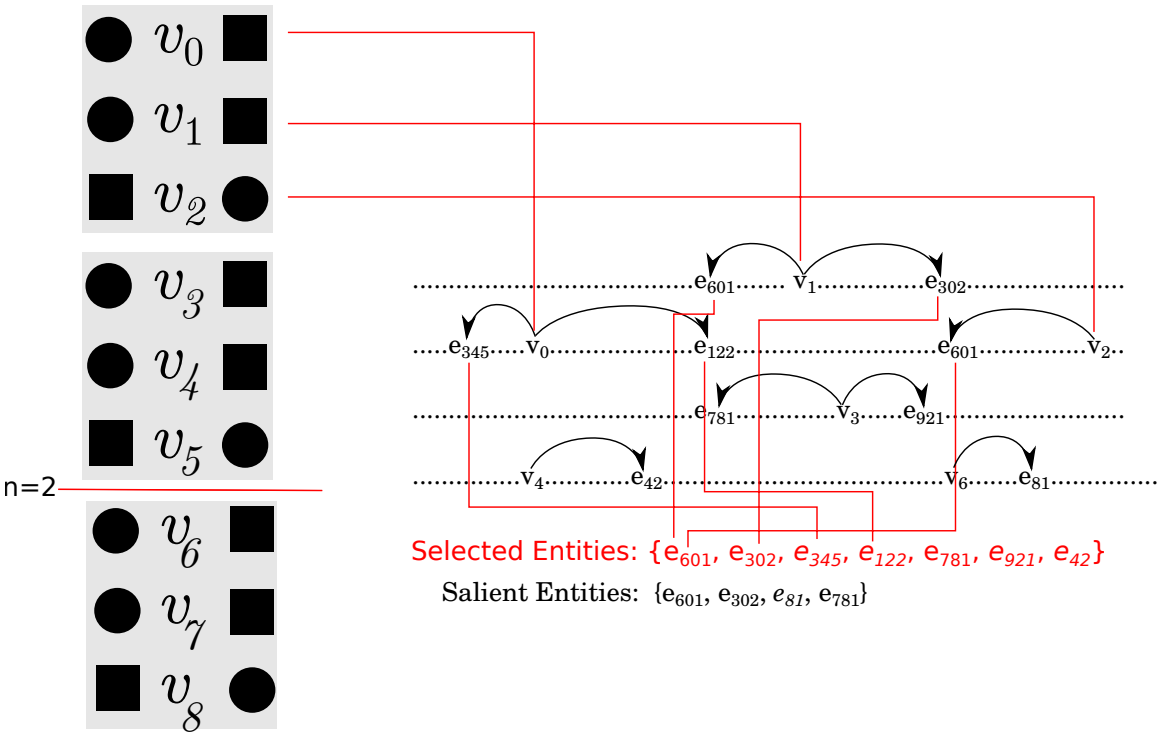


Figure 4.3: An illustration of the entity extraction process using schemas. In this particular case, the top two most present schemas out of the set of three were used for entity extraction. The whole process for the first schema is illustrated, though entities extracted by the second schema are also included in the selected entities set. Instances of the event verbs are selected by the schema, and arguments to the `SUBJ`, `OBJ` and `PREP` slots are added to the selected entity set. These are compared against the set of salient entities. Entities in italics are either False Positives (in the selected entity set) or False Negatives (in the salient entity set).

Bird et al. (2009) `wordpuncttokenizer` and is normalized for capitalization. Punctuation tokens are removed. Each entity extracted from the data is considered equal to the metadata entity if a fraction of the tokens r are equal between the two. This r value is set at 0.2, which is quite low, but justifiable, as any overlap between the open-class proper noun components likely indicates a match expressed differently from the normalized representation in the metadata: for example, an extraction of "Mr. Clinton" should match "William Jefferson Clinton" in the metadata. A higher threshold would have excluded these sorts of matches, which are typical of the writing style of the New York Times but differ in their metadata.

The fraction of entities from the metadata captured represents the *recall* while the fraction of things extracted actually found in the metadata indicates *precision*. NASTEA scores are reported as the F1 score of both of these values.

To validate these assignments and the low threshold of 0.2, I examined such assignments made in the test data set in 53 documents. In that subset, 368 assignments were made labelling entities extracted by in the NASTEA task as True Positives, False Positives, or False Negatives. Of those 368 assignments, 10 were incorrect. Two cases contained the target entity as a nominal complement: for example, "Suffolk County Tax Act" was extracted, but "Suffolk County" was the correct entity; similarly, the "Bruce Porter Company" was extracted, but "Bruce Porter" was correct. Only three cases seemed to confuse names: for example, "Joseph I. Liberman" for "Joseph L. Bruno." These, however, were only a small fraction of the total 368 assignments made.

4.3.2 NASTEA Curves and Their Interpretation

As much as I would wish for it to be the case, the most present schema does not always yield the correct entities. In many cases, adding additional schemas of high presence

is required. In the first pass over the `op1sem` subcorpus (see Sections 4.5.1 and 4.5.2), I use a set of schemas for each document, increasing this quantity by groups of five, starting at one. This allows us to see how well the first schema applied performed, followed by the the top 6, followed by the top 11, etc. If only the highest presence schema is applied, then that is expressed as "N_1;" for the top 6, that is reported as "N_6," etc. Nevertheless, N_1 results are of particular interest to us—this is the "I'm feeling lucky" narrative schema, the one with the highest presence with respect to a document. The N_1 performance should be highest in documents where canonicality most strongly applies.

I will discuss the results of the NASTEA task on sets of schemas generated jointly with the results of the MDL measures devised in the next section.

4.4 Evaluation: Minimum Description Length

Schank and Abelson (1977) first proposed scripts as a form of episodic memory. This definition was not merely meant as a theoretical cognitive or linguistic construct, but as something actually intended to both speed up and make more memory efficient information retrieval systems in computers. When given new descriptions of events, a Schankian system should be able to succinctly compress and store knowledge of the events contained therein, consuming and disposing of the original text.

If following the Schankian notion of script as a blueprint for our schemas, then they too should act as a good basis for an episodic memory, allowing us to interpret new sequences of events as succinctly as possible given prior representations of events. A more succinct representation is often the more insightful one as well, or more often than not enables the sort of deep, intuitive understanding from which insight blossoms.

One way to interpret this is to think of insight as a reduction of complexity of some space or set of information. An insightful explanation of a phenomenon or narrative reduces the amount of information required to deliver that narrative, given that some codification of the narrative structure captures details that are redundant or superfluous. In other words, the model with the *minimum description length* can be thought of as being maximally insightful.

Information theory has been drawn upon by many different fields, including text compression, and while the metaphor of compression is sometimes applied here, it is meant more in cognitive than raw computational terms, reflecting a thread of work beginning with G. Miller (1956)'s study of "chunking," where he famously determined that a typical person has a "channel capacity" of around 7. More so, G. Miller (1956) created a link between information theory and cognitive psychology, and it is in this sense that I talk about and seek to devise information theoretic measures of events and narrative.

That said, there is no particular reason MDL could or should be limited to events. Such measures have been applied to other aspects of language, such as the unsupervised learning of morphemes Goldsmith (2006). However, in this work, I am fundamentally interested in the unsupervised learning of narrative, so the atoms my model learns will be the atoms of narrative: events.

Previously, minimum description length has been used successfully in the unsupervised learning of morphemes (Creutz & Lagus, 2002, 2004; Goldsmith, 2006). Inspired primarily by Goldsmith (2006), I will define description length $DL(C, M)$ as follows:

$$DL(C, M) = |M| + |C|_M \tag{4.5}$$

$|M|$ is the size of the model and $|C|_M$, each of which I will define in detail below.

Since I am looking at narratives instead of morphemes, M will be a model of narrative—in this case, a set of narrative schemas. $|C|_M$ is the size of the corpus given a narrative model M.

In this section, I describe a narratological interpretation of $|M|$ and $|C|_M$. The definition of DL in Equation (4.5) will form the foundation of this evaluation metric.

4.4.1 Measuring Model Size: $|M|$

$|M|$ may be measured differently in different models. Generally, I define it as the number of nits[2]—bits, but with a natural logarithm base e rather than base 2—required to represent all *states* in the model. States are points within the language model that reflect specific *fragments* within a stream of continuous natural language. A fragment, in this sense, may be a string of tokens, or strings of tokens connected by some non-linear feature, such as a dependency.

Since this dissertation deals with narrative, I will focus on *narrative language states*—events and their participants. For example, in Pichotta and Mooney (2014; 2015)'s multi-argument tuples, each argument of each tuple originates in some narrative language state of the text.

To understand how this provides a meaningful information theoretic size measure for a set of schemas, I present the following example. Consider schema 669 from one of the generated sets, and fragment of text as containing events denoted within the schema in Figure (4.4).

Many of the events contained in this paragraph are represented as events in the schema. At least partially, the narrative can be represented by an encoding within

[2]While bits are often the default choice in natural language processing and information theoretic tasks, I have rebelled and chosen the nit. The nit is the natural unit of information; that is, one that requires fewer conversion factors in certain circumstances. Also, given that e is relatively close to 2, the difference between bits and nits is not shockingly different.

```
      669
0    B extradite C A
1    B detain    C A
2    B arrest    C A
3    B jail      C A
4    B release   C A
5    B deport    C A
```

> "Government policy is to deport (669) illegal entrants who are without refugee claims and to detain (669) the others, sometimes for years, while their pleas for asylum are examined (*) or contested (*) in the courts , to which such claimants have free access here. Although air detection and naval arrest (669) is catching (*) most illegal migrants before they land." — The New York Times[3]

Figure 4.4: A schema and related text for illustrating information theoretic size as applied to schemas.

the model: `deport`, `detain`, and `arrest`. To denote each of these states requires two dimensions of specification: first the schema, 669, then the events within the schema—5, 1, and 2.

Events aren't the only narrative states represented; there are also chains denoted therein. For example, "illegal entrants" in the text above are a narrative state that could be represented by the `OBJ` slot of deport—e.g. $(669, 2, 5)$, that is, schema 669, chain 2 (C), slot 5.

Given these two types of narrative states, I devise a measure of $|S|$ that is relative to the number of narrative states contained within it.

For a given schema, I define the entropy of S, $H(S)$ as:

$$H(S) = H(S_E) + H(S_C) \tag{4.6}$$

That is, the information contained in a schema is a combination of the entropy of the events and the entropy of the chains.

Furthermore:

$$H(S_E) = -\sum_{e \in S_E} p(e) \ln p(e) = \ln |S_E| \tag{4.7}$$

the reduction occurs here since all $e \in S_E$ are equally probable. This may seem strange, but this has to do with the set of schemas, not with the data that it should reflect. In measuring model size, I am creating a more theoretically motivated equivalent to "how many nits are necessary to describe this list of schemas." It's a description of the size of the schema model, not of the language data it was provided. Since each schema contains one and only one instance of each event, we know that each event is equally probable *within the context of each schema*. This says nothing about the data—intentionally. It is only a description of the model. Second, this is implied by the Chambers & Jurafsky (2009) definition of a schema as a tuple of a set of events and a list of chains, so the size of that is what is measured here. We could, of course, add some kind of probabilistic weighting to the events, but that is merely another piece of information that would increasing the size of the model. This is, fundamentally, a measure of how big and complex the set of schemas is, not necessarily a reflection of the data—that is what $|C|_M$ is.

A similar reduction occurs for the content of each chain, though the chains can vary in length, so no further reduction is possible:

$$H(S_C) = -\sum_{c \in S_C} \sum_{e \in c} p(e) \ln p(e) = \sum_{c \in S_C} \ln |c| \tag{4.8}$$

Thus, the whole set of schemas M as being of model size $|M|$:

$$|M| = \sum_{S \in M} H(S) = \sum_{S \in M} H(S_E) + H(S_C) = \sum_{S \in M} \left(\ln |S_E| + \sum_{c \in S_C} \ln |c| \right) \tag{4.9}$$

where $|S_E|$ is the number of events in the schema and $|c|$ is the length of chain c.

4.4.2 Size of a corpus given a model: $|C|_M$

Model size $|M|$ constitutes one aspect of a set of schemas we want optimized. In discussing an information theoretic measure for schema performance, we want to also want to consider how well a set of schemas can capture the events described in a set of holdout documents. Figure (4.5) illustrates the components that such a measure should include: effectively rewarding a set of schemas for capturing events in text and penalizing the performance score for events omitted by the text but contained in the schemas (E_{DS}) and vice versa (E_{SD}).

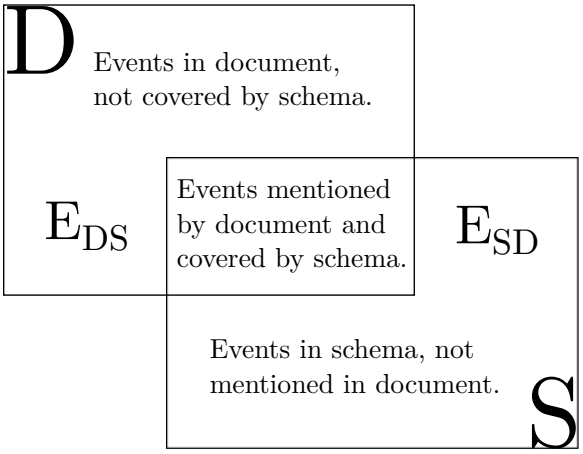


Figure 4.5: Illustration of the components in Formula (4.11).

In this section, I describe such a measure—how the size of the corpus $|C|$ is determined using some specific model M. The size of the corpus is a measure of the amount of information required to describe the events instantiated in a corpus of natural language. It is effectively an empirical measurement of the KL-divergence between the narrative model and the actual distribution of events denoted in the corpus C. In practice, this applies to a set of holdout documents, thereby measuring the model M's ability to encode previously unseen material that represents a random sample of its own training population.

I consider each document independently, e.g.

$$|C|_M = \frac{-1}{|C|} \sum_{D \in C} |D|_M \tag{4.10}$$

Note that this value is an average size per document—$|C|$ refers to the number of documents in the corpus. Thinking of the corpus size as an average rather than a total value is important—increasing the number of documents in holdout could bias the score in one direction or another with impunity. Instead, thinking on a per document basis prevents a hidden bias in the scores derived from the number of documents held out.

The schema best fitted to a document should capture as many pairs of events as possible that appear in that document. By default, this model assumes that all event pairs entailed by a schema are contained in a document when the schema is found to be the best match to apply to that document.

This also means that there are two types of error the model can make. The first type of error is where the schema overreached and predicted an event pair that did not appear in the document. This is penalized by the probability of the event in the schema. The second type of error is where the schema failed to predict an event pair that did appear. I penalize by the theoretical independent probability of the two events of the corpus. Since the model is being applied to holdout data, this guarantees coverage across novel event pairs.

Thus, I describe $|D|_M$ as:

$$|D|_M = \max_{S \in M} \left(\ln P(S) + \sum_{\kappa \in E_{SD}} \ln P(\kappa|S) \right) + \sum_{\kappa \in E_{DS}} \ln P(\kappa) \tag{4.11}$$

$\log P(S)$ is the information required to denote which schema has been applied to this document. $P(\kappa|S)$ is the penalty for each event pair in E_{SD}: the set of pairs that were predicted by the schema but missed (Type I errors). Every κ is a co-referring

argument pair, though where that pair comes from depends on which summation it is scoped under. In other words, Formula (4.11) expresses the information required to express that there was an exception to the expected sequence of events. $P(\kappa)$ represents the estimated independent probability of κ for κ contained in E_{DS}, the set of event pairs included in the document outside of the schema.

Unlike when computing $|M|$, I must now use actual probabilities. $\ln P(S)$ is essentially how many nits would be required to select a schema from the set of schemas and apply it to the document D. Then, since this specific schema S has been selected to "write down" the document, each pair of events in S are assumed to have happened in D. If some pair of events did not happen in D, those exceptions must be written down, which is what $\sum_{\kappa \in E_{SD}} \ln P(\kappa|S)$ indicates—the pairs of events the schema claims should have appeared in the document but did not actually see. $P(\kappa|S)$ is computed based on each of the event pairs in the schema—for a schema with 6 events, each missing pair would have a $P(\kappa|S) = 1/36$, or would require 3.6 nits to indicate.

For events contained in the document without representation in the schema, these must also be written down, but the schema model has failed to provide a means to do so. Instead, the MDL evaluation falls back to a "new" model. This new model—$\sum_{\kappa \in E_{DS}} \ln P(\kappa)$—uses the holdout data to estimate the probabilities of the event pairs in the corpus based on their independent probabilities. In other words, it comes up with a separate coding scheme by cheating, effectively, and tries to write down the event pairs in each document using that. In effect, this is a systematic way of applying penalties based on the data and consistent across the corpus. Each pair here's probability is computed with $P(\kappa)$. which is computed from the independent probabilities of both events as they appear in the holdout data. Previously unseen items are estimated as 1/the number of event verbs seen in holdout.

The independent probabilities are used to estimate somewhat inaccurate values—after all, the actual joint probabilities are likely a very good coding scheme for this. The point is not for the schema model to encode events perfectly; the point is to create a measure which can be balanced against model size and other measures.

Obviously, this is by no means a perfectly efficient encoding for the sequence of events. It is intended to, however, reflect a set of intuitions of what a schema represents and entails about a specific document it is applied to.

The probabilities have been transformed into ln space because, in some cases, the values approach zero too quickly, and the fragments are eventually rounded off.

In the next section, I deploy these measures—as well as the NASTEA task (Section 4.3)—on a set of narrative schemas derived from a subset of the New York Times corpus (Sandhaus, 2008a).

4.5 First Pass Experiment

In this section, I describe a first pass experiment where I evaluate a set of schemas derived from a subset of the NYT corpus (Sandhaus, 2008a) and juxtapose the results of both the NASTEA task and the MDL measures. These are the same schemas that were described and illustrated in Section (3.9).

The parameters used here were selected largely during early stages of this dissertation given repeated qualitative impressions of the schemas generated. A more formal selection procedure for parameters is employed in Section (4.7).

4.5.1 Data

I tested all algorithms and baselines on a subset of the New York Times Corpus.

Instead of using the whole NYT Corpus, I focused my experiments on a sub-corpus referred to as `op1-sem` throughout this dissertation. This was the first subset of documents pulled from the NYT corpus using a set of `online_producer` values as conditions for their inclusion, a subset which was chosen for their apparent semantic differences and similar quantities of documents:

Table 4.1: Counts of document categories selected from the `online_producer` tag for use in this study. Frequencies vary, but were chosen to be around the same order of magnitude and to represent different sorts of topics.

`online producer` category	counts
Law and Legislation	52110
Weddings and Engagements	51195
Crime and Criminals	50981
Education and Schools	50818
United States Armament and Defense	50642
Computers and the Internet	49413
Labor	46321
Top/News/Obituaries[4]	36360

The experiments in this chapter do not use the topic information in any explicit way, aside from the initial data selection. The next chapter (Chapter 5) explores the influence of topic distinctions in schema generation in depth.

All documents were pre-processed for POS tags, parsing, coreference, and NER. For NLP preprocessing, I used the Stanford CoreNLP suite of tools (Manning et al., 2014)[5] Primarily, CoreNLP was chosen for having both parsing and coreference facilities (de Marneffe et al., 2006; Lee et al., 2013) trained and readily useful. The full pipeline includes pre-requisites for those tasks, including but not limited to tokenization and part of speech tagging (Toutanova et al., 2003; Toutanova & Manning, 2000),

[5]Stanford CoreNLP, Version 3.4.1 (2014-08-27)

both of which are employed in typing chains and extracting events. The named entity annotations provided by the CoreNLP Pipeline (Finkel et al., 2005) are leveraged for determining types as discussed in some conditions (see Section 3.8.1).

4.5.2 Results

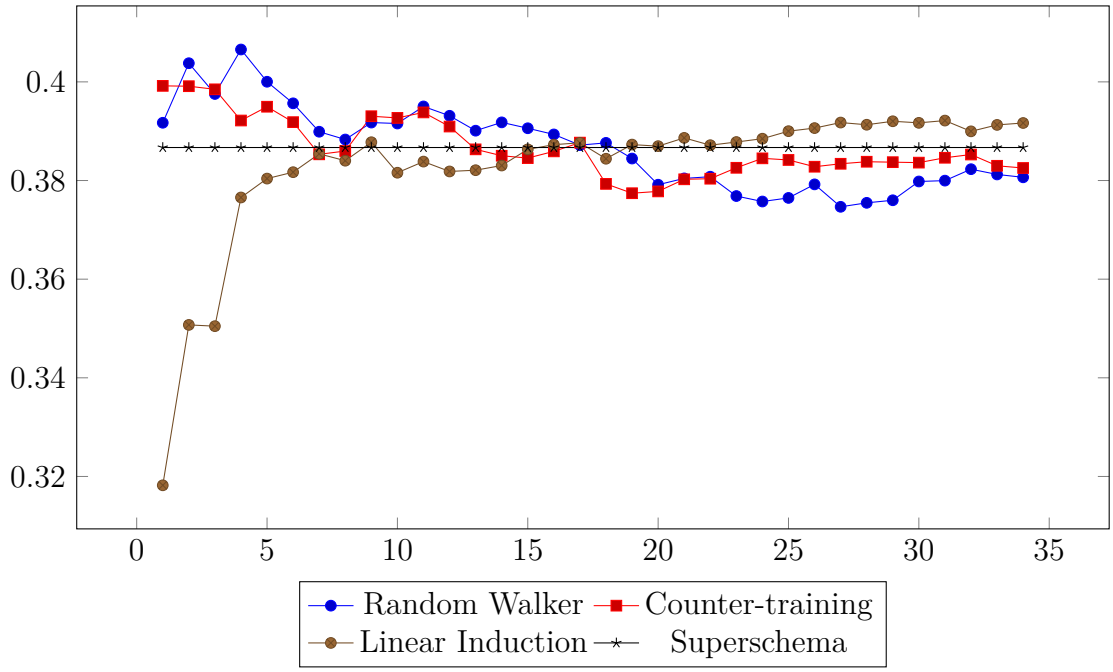


Figure 4.6: NASTEA curves for schemas generated by the germinators described in this chapter.

The results are contained in Table (4.2). The results are presented as components: $|C|_M$ and $|M|$ are presented separately to show the interchange of the components.

Clearly, some examples of a classic precision vs recall trade off can be seen between the two measures for different germinators. The two "strawman" baseline germinators clearly exchange between $|C|_M$ and $|M|$. For the single superschema model, $|C|_M$ was far larger than the other models. For the tiny schemas model, while its $|C|_M$ was on par with other schema models, it only did so by having an exceptionally massive $|M|$

value. Since larger MDL scores indicate worse performance, this indicates that the baselines perform about as poorly as expected in their respective domains.

Linear induction did relatively poorly on both metrics compared to the more sophisticated germinators: this is largely due to the massive number of schemas contained within its "tail." To compensate for this, the tail was hacked off. The linear induction-truncated mode features only the first 800 schemas generated by the germinator to put it on common ground.

Counter-training and random walker both outperformed even the linear induction-truncated schemas. Counter-training outperforms the linear induction-truncated in both metrics, though just barely attaining a better $|C|_M$ value than the linear induction-truncated schemas. The random walker outperforms only in $|C|_M$, but does so with a larger $|M|$.

Table 4.2: Combined NYT `op1-sem` Results for NASTEA and MDL, reported to four significant figures. † indicates, in the $|C|_M$ cases, that this value was significantly different ($p < 0.001$) from all other germinators tested in a paired t-test for its precursor $|D|_M$ values.

	N_1	N_6	MDL $\|C\|_M$	MDL $\|M\|$
Tiny Schemas			36360	4261000
Superschema	0.387	0.387	$\dagger 5.772 \times 10^9$	39.69
Linear Induction	0.318	0.381	35980	14080
Counter-Training	0.399	0.391	36050	4410
Random Walker	0.391	0.396	† 35010	5369

4.5.3 Discussion

First, between the baseline models and the "true" schema models, it seems the baselines performed worse than the schema models in most circumstances, though only the random walker reached a degree of improvement enough for significance in a paired

t-test of the sequences of $|D|_M$ values. There are exceptions to this generalization, however. The first is the N_1 performance of the schemas is relatively poor compared to the superschema baseline—however, applying the next five schemas in rank typically pushes them above the superschema baseline score. This suggests the documents may not be as homogenous as our models here imply. The model itself is an entity-based evaluation focused around proper noun phrases, lacking gold coref. However, it thus far suggests that perhaps documents are not best described as instantiations of single schemas. This is explored in more depth in Chapter (5).

With respect to the MDL measures, the schema models themselves generally outperform the baselines in all $|C|_M$ and $|M|$. The `tiny schemas` baseline did remarkably well—insignificantly different from the other schemas—but at a high $|M|$ cost; conversely, as expected, the `superschema` baseline $|M|$ has a small model size—essentially being a list of verbs—but struggles to represent the corpus, resulting in an indefensibly massive value for $|C|_M$.

While the measures can be considered in isolation, considering them as a whole paints a better picture of performance. No single measure of this collection can be considered in isolation. NASTEA reflects the schemas' ability to reflect the intuitions of annotators of salient entities, and the two MDL measures reflect abstract ideas of the utility of the schemas—that is, one expects a useful model to both provide a compact encoding of unseen events ($|C|_M$) and to be concise in itself ($|M|$). The superschema baseline, for example, gives a strong NASTEA performance, and an extremely small $|M|$ value, but this is revealed to be done at the expense of the $|C|_M$ value, which is massive. For sake of completeness, I attempted to run the tiny schemas baseline could not even be run through NASTEA, but this proved too slow—the massive model size simply took too long to traverse in a reasonable amount of time.

However, we see from the schema measures a balanced performance in all measures, attaining improved performance in all circumstances with no specific measure reaching an extreme. With the goal in mind of using schemas for unsupervised analysis, balance between these measures is what really counts. The schemas generated through actual germination have balanced scores that reflect their own tendency to represent pieces that are easier to interpret; the imbalanced MDL measures for the baselines reflect their inability to provide interpretable components, by design reflecting inabilities to meet the intuitions driving the development of the pair of MDL measures.

Within the schema-based models themselves, the counter-training model seems to have performed the best in terms of $|M|$, while the random walker performed better with respect to $|C|_M$, albeit at a higher $|M|$ cost.

4.6 Contextualizing Baselines for NASTEA

In addition to running baseline-style schemas through NASTEA, I also established a number of baselines to better contextualize the salient entity extraction that occurs in NASTEA.

The first of these baselines, **All NNP**, grabs all noun phrases that contain a token tagged as a proper noun from a given document. The second, **NNP+NER**, is the same as **All NNP** except that it filters out noun phrases that were not tagged by the Stanford CoreNLP NER as a `Person`, `Organization`, or `Location`.

Furthermore, NNPs can be filtered by frequency instead of selecting them all blindly. This I refer to as **NNP @ n**, meaning the n most frequent noun phrases in a given document. Of particular interest is **NNP @ 5**, since that is roughly the average number marked as salient in each document, technically something we could

ascertain from the training data, and **NNP @ 2**, since that gives us a simple case of a protagonist-antagonist pair.[6]

The final baseline is a little different from the others—**DEP**, which uses the dependency annotations from CoreNLP, like our schemas do. From every verb in the document, the **DEP** baseline gathers any NNPs that appear in what our schema system would consider a `SUBJ`, `OBJ`, or `PREP` slot. This is arguably the most difficult baseline to beat as it leverages much of the same data available to the schemas, but beating it with a schema driven system shows that the schemas contribute something to the identification of salient entities.

Table 4.3: Contextualizing baselines for NASTEA.

Baseline	F_1 on Dev	F_1 on Test
All NNP	0.319	0.321
NNP+NER	0.127	0.127
NNP @ 1	0.250	0.249
NNP @ 2	0.332	0.334
NNP @ 5	0.379	0.382
DEP	0.403	0.405

From these baselines, it is clear that the schemas do contribute something to the extraction process, and that something is more than mere filtering through a handful of select dependencies.

4.7 Parameter Climb for Optimal NASTEA Performance

To more rigorously assess the value of these results, I further divided the `op1-sem` corpus into development and test components, each 10% of the corpus, and then

[6] And Dr. Chambers requested it, which was easy enough to oblige.

conducted a parameter climb using each of the germinators discussed earlier in this chapter.

The five parameters selected for the climb were, with default values indicated in parentheses, in the order that each was optimized:

- $\beta(1.0)$: indicating whether chains should be separate or not. When β is lower, slots are more often linked into chains, and events are split into new schemas more often in the LI germinator.

- $\lambda(1.0)$: indicating relative weight between raw event *pmi* values and scores for type counts. When λ is high, slots are more likely to be linked because of their associations show a strong affinity for a specific chain type (e.g. `«search, SUBJ>, <detain, SUBJ»` sharing "officer" as a type is more important than the bare fact that they appeared together with a high *pmi*.)

- $c_{\min}(1)$:[7] minimum counts of a verb to allow it to be included in schema induction. In other words, if the verb "burnover" appears only once, it is removed from all consideration.

- $s(6)$: maximum schema size. Once a schema is equal to this size, its growth stops.

- $S(100)$: maximum number of schemas (irrelevant for the linear induction germinator).

Each parameter was optimized once. For β and λ, these were tested in near order-of-magnitude increments: 0.1, 0.5, 1.0, 2.0, 10.0. $c_{\min}$ was tested from 1 to 7. s was

[7]This defaulted to one because in many of the early experiments of this dissertation, it was not clear how much data exactly I would be able to work with. Consequentially, I added it as a parameter here.

tested from 2 to 12, excluding 9 and 11 for a speed boost. S was tested from 100 to 800 in steps of 100.

s and S were set low and optimized last because increasing the number of schemas and events greatly increases the number of computations required, greatly slowing down the germination process. β and λ were done first as more obvious "parameters," and $c_{\min}$ was done before the last two parameters to potentially remove some events from consideration, thereby adding a speed boost before running with larger s and S values.

Schema size and number of schemas were both computed last since those potentially involved drastically increasing the number of computations performed—e.g. the jump from 100 to 200 schemas alone doubles the amount of work that needs to be done. Similarly, increases in schema size can result in a great many more comparisons for each new event added.

During each step, all N_n values for the NASTEA task were optimized from 1 to 50 with a step of 1.

To speed up evaluations, I randomly sampled 1/324 hold-out documents in both the development and holdout portions. The random number generator was seeded with a fixed integer value during the dev and test phases of the experiment.[8]

In some cases, two parameter values resulted in a tie. By default, I chose the value that would most likely result in faster germination down the line. However, if that were not the case, I chose the value that achieved tie score by applying the fewest schemas to solve the problem (e.g. with the lowest n value in N_n).

[8]This integer was 19800518—used in any random sampling in this dissertation—and was chosen because it is a concatenation of the digits of the date of the death of Ian Curtis.

4.7.1 Parameter Climb Results

The results for the Dev steps of the parameter climb can be seen in Figures (4.7 - 4.11). Some parameters seemed to hint at some sort of underlying curve; however, it's hard to tell from so few points whether any sort of relationship holds. Most parameter settings retained some degree of improvement over the DEP baseline (Section 4.6), albeit small, though rarely falling below.

The results on the Test set of documents can be seen in Table (4.4). All schemas dropped in performance. For the random walker and linear induction schemas, scores dropped substantially from the Dev to Test step. The linear induction schemas even did worse than the DEP baseline. However, the counter-trained schemas dropped only 0.001 from Dev to Test; despite scoring the most modestly on the Dev step, counter-training outperformed the other two germinators in the Test step.

Table 4.4: TEST Results, using the n value from the last step applied in the parameter climb.

	Dev N_n	Test N_n	n	λ	β	c	s	S
CT	0.416	0.415	22	1.0	2.0	3	5	100
RW	0.422	0.407	8	0.5	1.0	5	12	100
LI	0.419	0.395	6	1.0	10.0	3	4	∞

Surprisingly, generating only 100 schemas produced the best results for the counter-training and random walker germinators. It is possible, however, that these values peaked here because the number of schemas was the last parameter tested, so they were optimized before reaching this step exclusively to this value. To resolve this would require a more thorough search of the parameter space.

Some aberrations occurred during the random walker germination climb worth documenting. During the $\lambda = 10$ test of the random walker germinator, an overflow

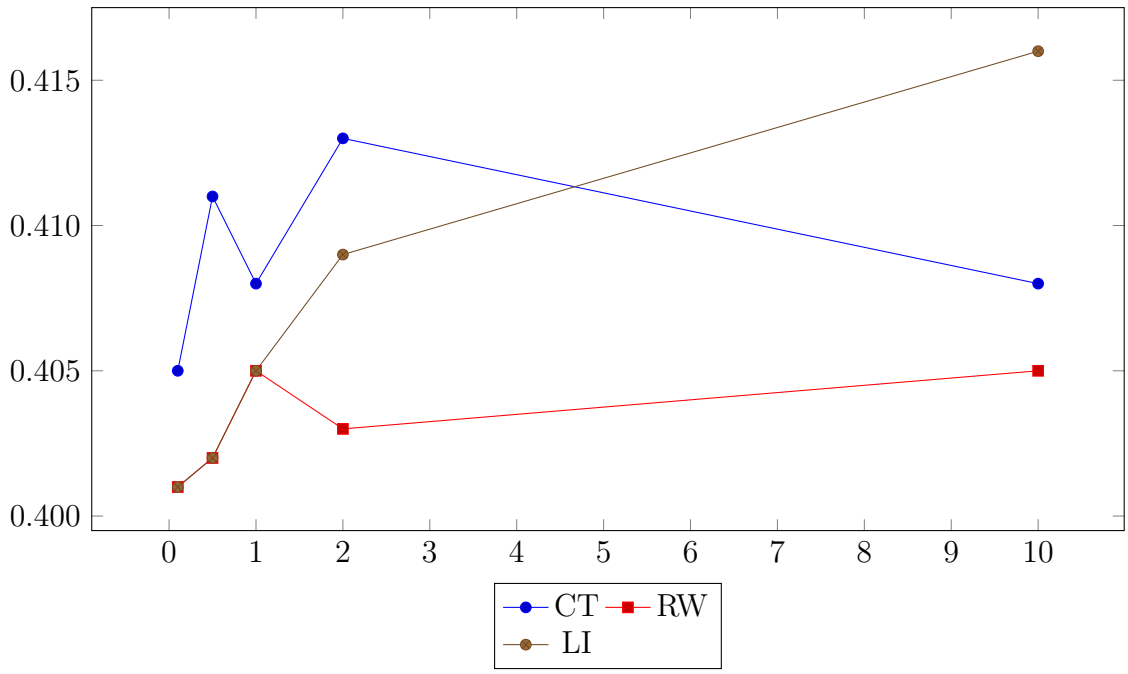


Figure 4.7: Plot of β values tested during hill-climb. The x-axis indicates values tested. The y-axis indicates F1 score on the NASTEA task used at that value. Each germinator is indicated by a different series.

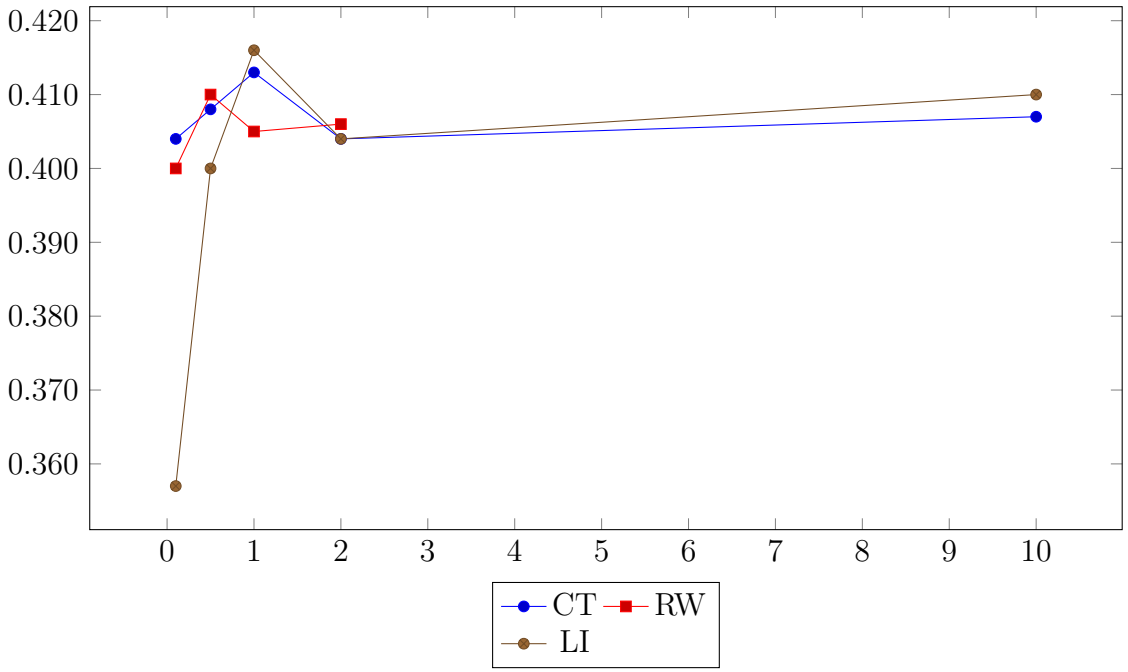


Figure 4.8: Plot of λ values tested during hill-climb. The x-axis indicates values tested. The y-axis indicates F1 score on the NASTEA task used at that value. Each germinator is indicated by a different series.

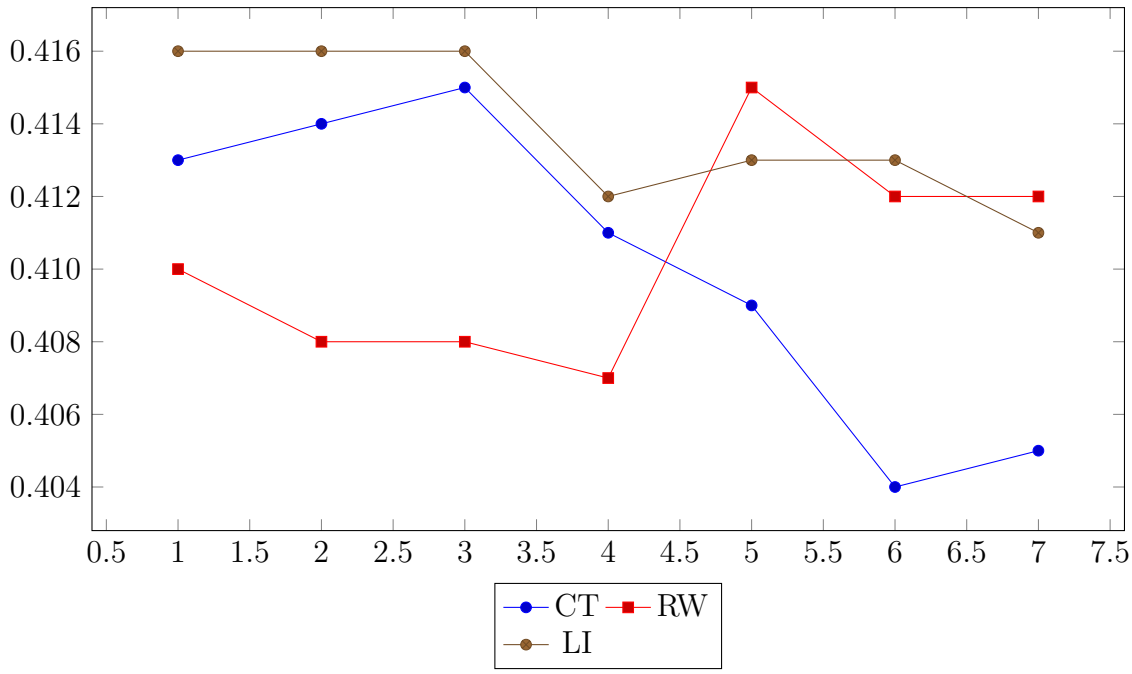


Figure 4.9: Plot of counts-min c values tested during hill-climb. The x-axis indicates values tested. The y-axis indicates F1 score on the NASTEA task used at that value. Each germinator is indicated by a different series.

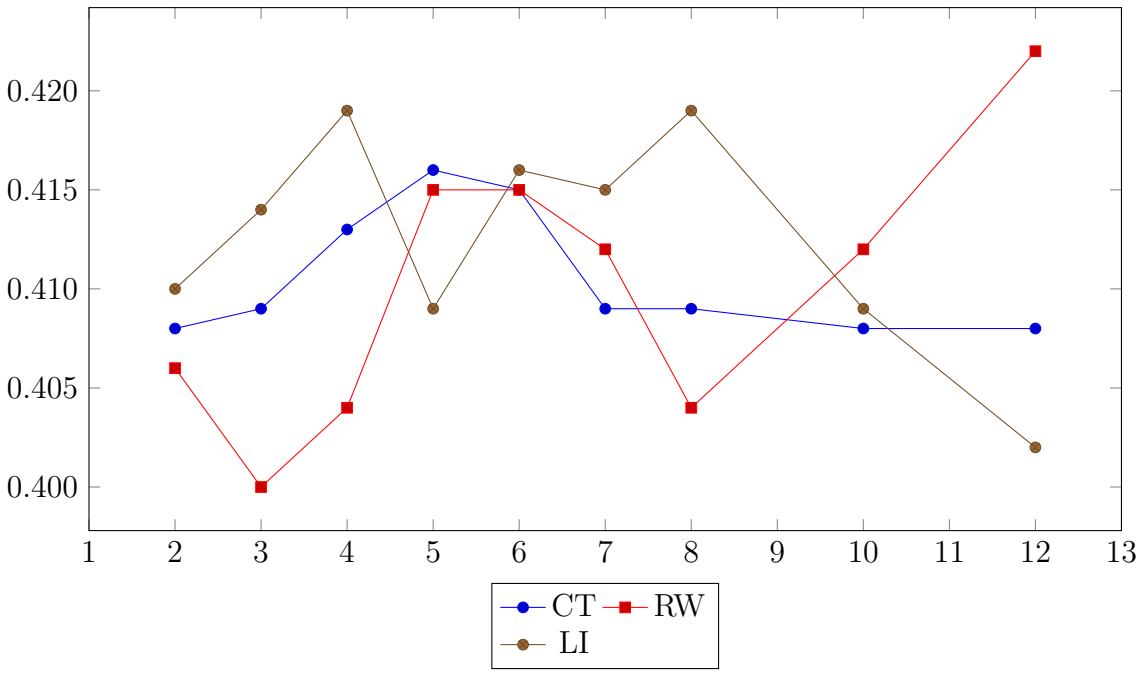


Figure 4.10: Plot of schema-size s values tested during hill-climb. The x-axis indicates values tested. The y-axis indicates F1 score on the NASTEA task used at that value. Each germinator is indicated by a different series.

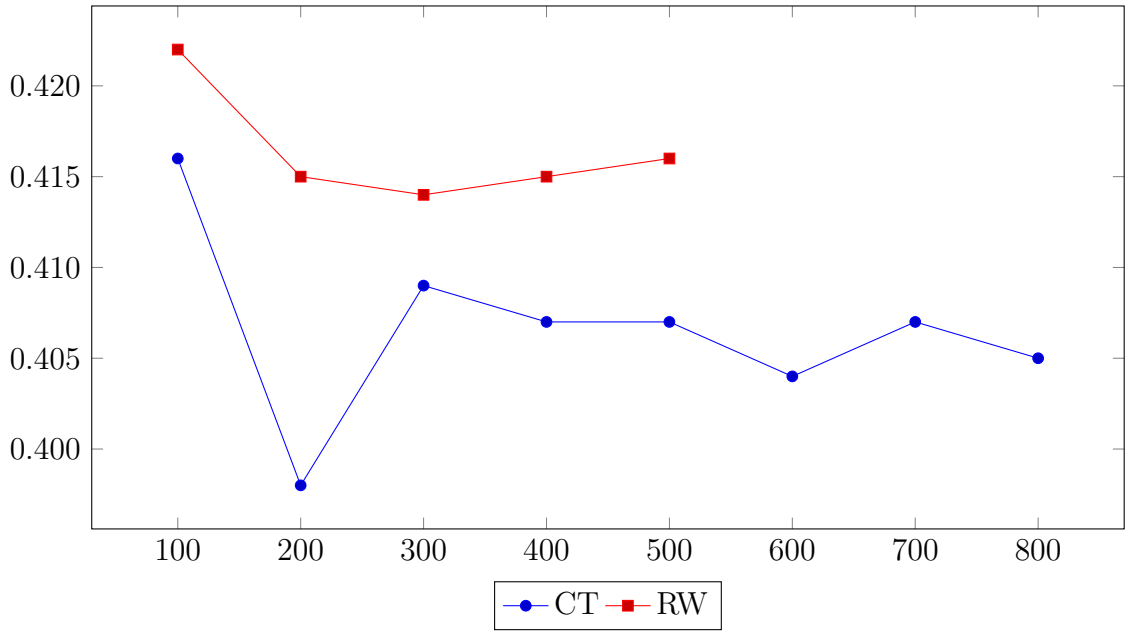


Figure 4.11: Plot of number-of-schemas S values tested during hill-climb. The x-axis indicates values tested. The y-axis indicates F1 score on the NASTEA task used at that value. Each germinator is indicated by a different series.

error caused schema germination to crash. This was not resolved. During the max number of schemas step, because the random walker schemas were most effective at size 12, the germinator leaked a lot of memory during the next phase of the parameter climb, the maximum number of schemas climb. As a result, $S = 600, 700, 800$ were skipped during this portion of the random walker climb.

Ties occurred twice during the parameter climb, both while improving the linear induction germinator. The first was during the min counts c step where counts 1, 2, and 3 obtained scores of 0.416. I selected $c = 3$ to eliminate more verbs and improve speed. In the next step, the max schema size step, $s = 4, 8$ tied. I selected 4 because it did so at a lower n value, 6, as opposed to 11 for $s = 8$.

4.7.2 Discussion

Overall, the parameters seem to at most contribute small improvements to the performance of schemas with respect to the NASTEA task, creating changes around 1 - 2% in the scores. However, although these are small improvements, they make a big difference with respect to the highest baseline score, **DEP**, which sits at around 0.403. Most results during the dev phase stayed above this line, but in some circumstances wandered quite close or below this line. It is worth keeping in mind that the **DEP** baseline is quite difficult to beat, as it uses the very same `SUBJ`, `OBJ`, and `PREP` relationships used by the dependencies. The schemas act as a filter on the `DEP` baseline, removing events not included in the most highly present schemas from consideration.

It seems that no specific parameter has a particularly strong trend for all germinators, but some pairs of relationships are worth pointing out. These may require more sophisticated analysis to tease out whether these relationships are significant and reliable. Nevertheless, they are worth commenting on.

With respect to β, most germinators generally trended upward. This is surprising. CT and RW germinators use β strictly as a parameter for creating chains within schemas, informing the decision to join slots from a newly included event. β, in addition to deciding how chains should link together internally, informs LI whether or not a new schema should be created. A low β means that new events will be added to almost any schema that has ever appeared with another. This means that the pool of actively generated schemas will never become large enough to approach any sort of global maximization since new events will likely be inserted into some schema in the current, small pool—schemas which then quickly fill with poorly fitting events. Higher β forces the LI germinator hold out longer and create more new schemas. LI trended with RW almost exactly for the first three parameter values, and also followed

a similar upward trend for CT, but then LI outperformed all other germinators at $\beta = 10$, though, so this more conservative parameter setting seems to have benefited LI's schemas.

With respect to λ, most germinators seemed to create a small curve between 0.400 and 0.42, peaking at a parameter setting of 1 (except RW, which peaked at 0.5, and the run with $\lambda = 10$ did not successfully complete). λ expresses the weight between the raw *pmi* component of *sim* and the typed component. A lower λ means lower weight for chain types; a higher λ means that types play a more important role than shared slots alone. LI seemed to perform extremely poorly at $\lambda = 0.1$. At this parameter setting, typed chains virtually have no effect, leaving the *pmi* component of *sim* dominant; that is, most of what matters in that condition is whether two verbs shared a coreferring argument through their slots and not that those slots shared similar argument types. In other words, typed chains played a crucial role in linear induction's performance, reaching the highest value when both components were equally balanced at $\lambda = 1.0$.

For the later items in the climb, there are definitely caveats for the scores—for example, the other parameters have been searched within the context of the first parameters, so they are already optimized for the default values. This means that improving from that point or near that point is unlikely since everything done up to this point has been optimized for these defaults.

With respect to counts-min c, the scores do seem to largely stabilize, at this point only varying within a 1.2% range. However, most germinators tend to trend downward with increasing c, with some notable exceptions. For $c \leq 3$, CT and LI increase or remain flat, afterward dropping. RW trends downward or flat until reaching $c = 5$, wherein the score shoots up from the lowest to highest value it obtains in the climb. I suspect this is because of the random nature of the RW germinator—it needs to

eliminate more odd possibilities because there remains some random chance that it selects something in its long tail to add to a schema. Removing minimum counts—more than the other germinators, protects the RW germinator against dipping into this tail of rare events and gives it a more robust set of items to randomly draw from. This does not totally explain why things shoot up around $c = 5$, and this might be overtraining taking place rather than any specific result.

With respect to schema size s, there seems to be a curve early on peaking around 5. LI oddly drops to a low score at five, and RW peaks in performance at $s = 12$. This, again, might be the random nature of the germinator, giving it more opportunities to make better picks and a few strange ones. CT clearly has the smoothest curve here, which may reflect its robust performance on the test set.

With respect to the final parameter, number of schemas S, performance surprisingly drops off as more schemas are added. This seems to contradict conventional wisdom, that having more in the model is necessarily better. It could be that all parameters up to this point have been optimized for $S = 100$ by default, a choice made for speed more than anything else. LI, notably, does not have this parameter.

With respect to the test results, the CT schemas seemed to perform most robustly, obtaining the highest test result despite obtaining the lowest dev result. RW dropped 1.4% in test, and LI dropped nearly 2.4% below dev, crossing below the 0.405 performance of the **DEP** baseline on test data. This definitely is in counter-training's favor, as it seems to indicate robust performance.

While here I sought to improve schemas by improving parameter settings, what if I provide their germinators with better data? I address this in the next section.

4.8 OntoNotes Experiments

While the NYT results are generally positive, an evaluation of our evaluation metrics would provide more confidence in those results. Given the imperfect nature of automatic annotations, one way to do this is to provide higher quality training data to the system overall, if available. A reliable evaluation metric—reflective of the linguistic realities of the content it intends to model, interpret, or understand—should reflect the contributions of these more reliable annotations and thus show improvements when it is given improved data. In other words, with the old adage "garbage in, garbage out" in mind, "better data in, better results out" should hold true for a good evaluation.

To this end, I have also conducted similar experiments to those presented above but using the newswire portion of the OntoNotes corpus (Weischedel et al., 2013), in particular, the portion annotated with coreference information, just over 900 documents. The goal in this experiment is to determine the effect, if any, of gold standard coreference information on the quality of the schemas produced at the end of the pipeline. In this experiment, I produce two sets of schemas: one from Weischedel et al. (2013)'s gold standard OntoNotes annotations and another from automatic annotations using CoreNLP (Manning et al., 2014).

Experimenting on OntoNotes provides some distinct advantages and disadvantages. The syntactic trees and coreference information in OntoNotes are hand-annotated and gold standard, so errors in schema extraction cannot be blamed on the coreference resolution system and parser. On the other hand, the set of documents is quite small and the set of documents is quite broad, from many different news sources, which may or may not affect the quality of schemas generated. OntoNotes

also lacks salient entity annotations essential for the NASTEA task, so the NASTEA task cannot be done on this corpus.

Given the smaller quantity of data, I lowered the threshold length for a coreference chain length to be used to generate schemas from **5 coreferring mentions to 3 mentions**. This was to reduce data sparsity given the relatively small size of OntoNotes.

4.8.1 Results

The resulting schemas were evaluated on both components of the MDL measure. The schemas themselves and alone directly determine $|M|$; $|C|_M$ depends on the set of holdout documents—in this case, the same holdout documents as specified in Pradhan et al. (2011).

The results of the OntoNotes experiments are shown in Table (4.6). On the cloze task, the model built on the automatic annotations outperformed the gold standard annotations, but only marginally so. Only the superschema baseline was significantly different from the rest of the results in a paired t-test of document size ($|D|_M$) values. A paired t-test of the $|C|_M$ values did not reach significance either.

Within each experiment, both on the automatic and gold standard results, the schema-based approaches generally failed to show any improvement over the tiny schemas baseline with respect to the corpus size ($|C|_M$) score, the only exception being LI schemas in the automatic experiment.

Note that without any sort of "salient" entity annotations, NASTEA was not performed on the OntoNotes corpus.

Table 4.5: Combined OntoNotes Results for MDL on OntoNotes' gold standard coreference and parses. Linear induction germination generated fewer than 800 schemas, the number generated by the other two germinators. Superschema $|D|_M$ values were significantly different from the others ($p < 0.001$, marked with †). No other pair of scores in tests obtained significance.

Cloze (Average Rank)	Gold 1020		Automatic 993	
	MDL $\|C\|_M$	MDL $\|M\|$	MDL $\|C\|_M$	MDL $\|M\|$
Tiny Schemas	18750	16800	18800	23270
Superschema	†13430000	21.77	†17170000	22.75
Linear Induction	18530	867.9	18560	1283
Counter-Training	18820	3065	18760	3337
Random Walker	18780	3154	19070	3332

4.8.2 Confusion Matrices

As I did in Section (3.9.2), I illustrate the similarities between the schemas in this section with confusion matrices.

Table 4.6: Fuzzy Jaccard similarities between both sets.

	ON to ONA	ONA to ON
LI	0.162	0.161
CT	0.143	0.153
RW	0.142	0.155

The OntoNotes self-similarity matrices are quite similar to those derived from the NYT corpus, but with some notable differences.

With respect to linear-induction schemas, the central diagonal is not a stark white, but fades to gray as we go down the line. The gray scaling is proportional to number

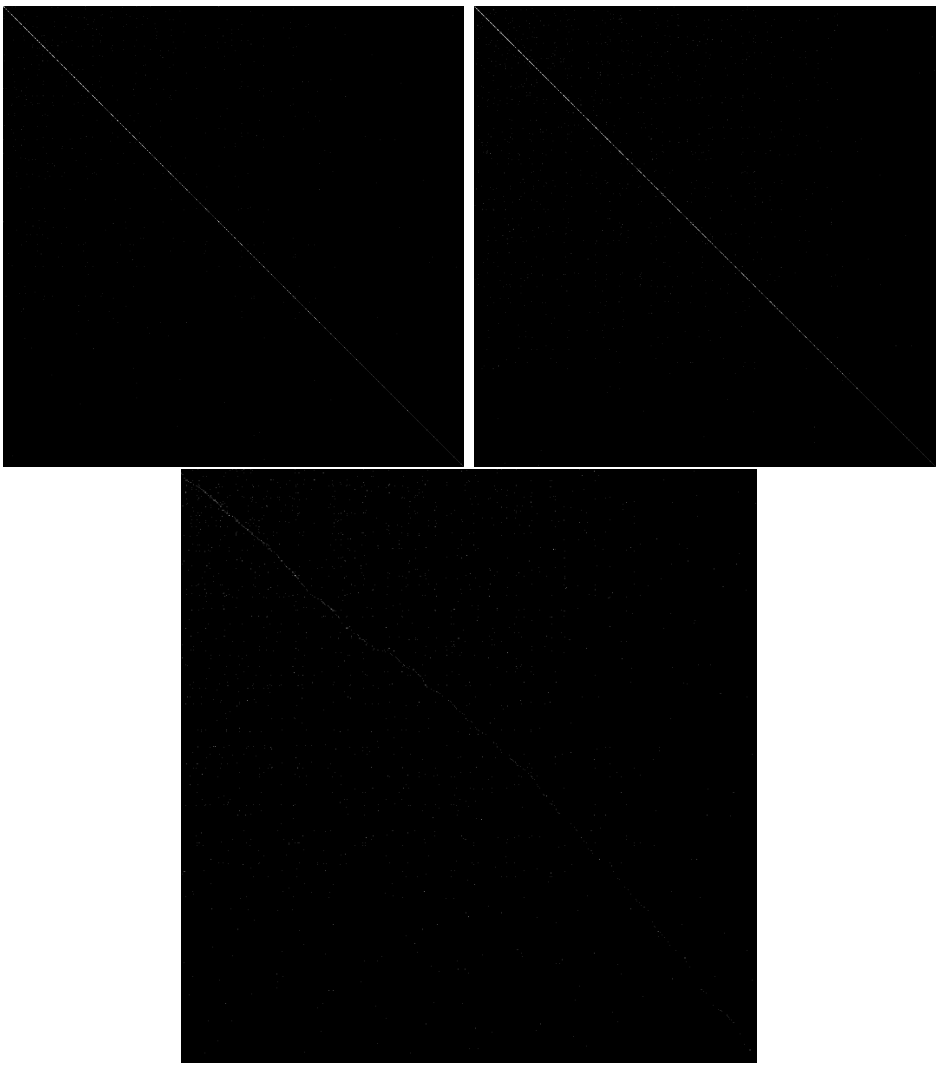


Figure 4.12: Linear induction (truncated) self-similarity matrices. Schemas from the gold standard data are on the left; schemas from the automatic data are on the right. Note that because there are fewer documents, fewer schemas total were generated. These matrices are based on the raw counts of similar events, so the diagonal appears to fade as the schemas diminish in size.

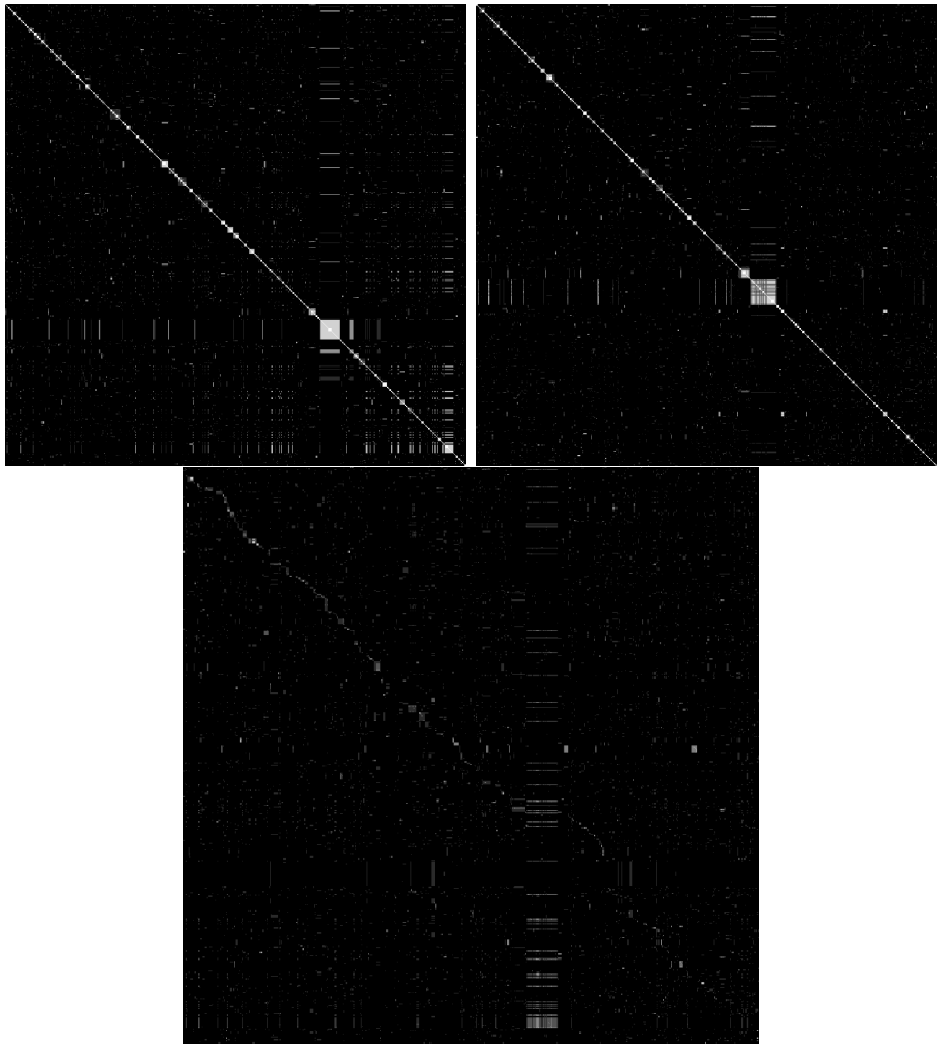


Figure 4.13: Counter-training (truncated) self-similarity matrices. Schemas from the gold standard data are on the left; schemas from the automatic data are on the right.

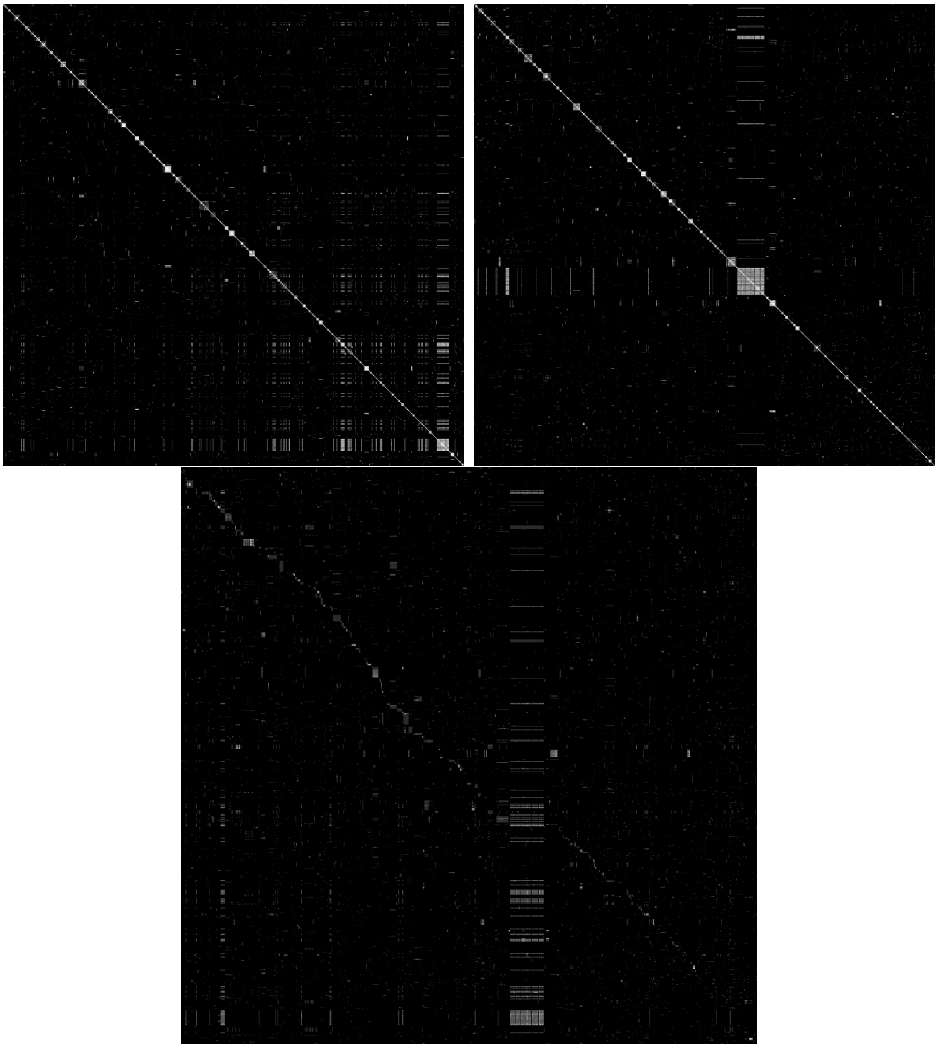


Figure 4.14: Random walker (truncated) self-similarity matrices. Schemas from the gold standard data are on the left; schemas from the automatic data are on the right.

of similar events, not to the proportion of similar events, which white being the maximum value. What we see here is a product of many of the schemas not reaching size 6. Since the portion of OntoNotes here is much smaller than the portion of the NYT corpus used, there were simply fewer candidate event verbs to fill out and yield more schemas. The linear induction automatic schemas show visibly more similarities across different members of the set, visible by more non-black pixels outside of the diagonal.

Counter-training seems about the same; however, given the gold standard to automatic comparison, while many schemas seem to share members, they do not match up exactly in most circumstances, and the "bright blocs" share nothing in common.

The random walker schemas here more closely resemble the counter-training confusion matrices from the NYT corpus than their similarly generated counterparts, with a number of tight, large blocs hugging the diagonal. This may be a consequence of having less data. The RW Superbloc (Section 3.9.3) was given more opportunities to grow with NYT data, so the generic sort of events contained in that bloc had opportunities to be exposed to more aberrations and combinations of verbs, exposing those schemas to more possible évents to tap into during the random walker's search of the space. However, there is simply less OntoNotes data, so there are fewer opportunities to obtain new permutations of combinations of generic event verbs with other event verbs.

Overall, the internal similarities and cross-annotation similarities are about what is expected given new data.

4.8.3 Discussion

These results are surprising. The gold standard data from OntoNotes provides high quality, hand-curated annotations, yet produces no significant improvement with

respect to $|C|_M$ compared to the automatic annotations, with the only exception being the superschema baseline, whose $|C|_M$ measure improves by about 22% when provided gold standard annotations. A minor improvement in cloze scores also reflected this same improvement seen in the $|C|_M$ value.

Model sizes do improve, though. The gold standard annotations yield models that were on average 15% smaller, a peak improvement of 32% between the linear induction schemas, with little difference between the superschema baseline and the random walker. This means, on the positive side, that the gold standard annotations are able to encode the holdout data for about the same information cost with a smaller model.

Nevertheless, while there is no significant difference between the gold standard and automatic annotations, this must be explained. After all, if feeding the system better data, one would expect better results. What accounts for this lack of improvement or, in some cases, insignificant degradation?

The most immediate difference between the two models can be seen as a property of some of the precursor components. Before schemas are generated, a number of steps occur, including a step containing only CAPs extracted from documents, not counted or reduced in any way. The gold standard annotations extracted tuple file is 865.0 KB, while the automatic annotations produced a file of 1,091.0 KB. Simply put, the automatic annotations yield more content for the model. The question here is whether it's this difference that resulted in the difference in scores.

However, to the systems described in this dissertation, there are assumptions about what constitutes a narrative which must be present to be leveraged. To be selected as a possible participant in a schema, a coreference chain must be present, of a suitable length (in this experiment, reduced to 3), and must be in an appropriate dependency relationship with a verb, which is how a CAP is selected from the

annotations of a document. Looking at the content of the intermediary file described above, the gold standard coreference annotations produced an average of 17.4 CAPs per document; the automatic annotations produced an average of 24.0 CAPs per document.

Consider this entire OntoNotes article, both with gold standard and automatic annotations:

> "Sharp increases in the price of fresh produce caused <0>Spain's <2>September consumer price index <1>to shoot up 1.1% from the previous month, pushing the annual rate of inflation to 6.8%, the National Institute of Statistics said Friday. <1>The monthly increase is the highest recorded in the past four years. <0>The index, which registered 156.8 at the end of <2>September, has a base of 100 set in 1983 and isn't seasonally adjusted. Prices have risen 5.9% in the first nine months of the year, outstripping both the initial 3% inflation goal set by the government of Socialist Prime Minister Felipe Gonzalez and the second, revised goal of 5.8%." — OntoNotes, `WSJ 2389`, Annotations from OntoNotes

In the gold annotation here, none of these coreference chains are long enough to contribute to schema generation, given the length 3 threshold. Even were I to lower the threshold to 2, most mentions would not contribute anyway, given the assumptions used in this system. Chain 0 appears as the `OBJ` of *cause* and the `SUBJ` of *to shoot up*, but the second mention appears in the `SUBJ` slot of a the stop word *have*, and so therefore would be ignored. Chain 1 coreferences a verb and a nominalized mention o the same event as the complement of a copula, both which are not covered by the assumptions generally held in schema generation. Chain 2 does not appear at all in

a direct hierarchical relationship with a verb. Thus, only one co-referring argument pair would be extracted from the whole article: ⟨⟨cause, OBJ⟩, ⟨shoot, SUBJ⟩⟩, with the type *index* attributed to the whole chain.

We see similar patterns and trends in the automatic annotations:

> "Sharp increases in the price of fresh produce caused <0>Spain's September consumer price index to shoot up 1.1% from the previous month, pushing the annual rate of inflation to 6.8%, the National Institute of Statistics said Friday. <1>The monthly increase is <1> the highest recorded in the past four years. <0>The index, which registered 156.8 at the end of September, has a base of 100 set in 1983 and isn't seasonally adjusted. Prices have risen 5.9% in the first nine months of the year, outstripping both the initial 3% inflation goal set by the government of Socialist Prime Minister Felipe Gonzalez and the second, revised goal of 5.8%." — OntoNotes, WSJ 2389, Annotations from CoreNLP

The results are exactly the same here. Chain 0 is almost identical, excluding the appositive following *the index* but otherwise yielding precisely the same CAP as in the gold data. Chain 1, a copula, provides no CAPs.

In both cases, neither article contains a coreference chain long enough or with the appropriately assumed relationships to capture more than one single coreferring argument pair from the source material, and this pair is itself the same in both articles. Thus, given the assumption on which the systems described in this dissertation are built, it lacks any sort of protagonist which schema and script-based language systems expect and attempt to leverage (Chambers & Jurafsky, 2008, 2009). Many articles are of this length—or even shorter—in the OntoNotes corpus.

However, this does not totally address the issue yet of whether and how the automatic annotations ended up performing slightly better with respect $|C|_M$ over the gold annotations, since here, despite having different annotations, the result was identical. So, I took a look at one of the longer documents in the corpus, `WSJ 1018`, and highlighted coreference mentions that were in one of the dependency slots we were interested in, either `SUBJ`, `OBJ`, or `PREP`:

> "Rupert Murdoch acquired a 25% stake in Grupo Zeta S.A. , the leading Spanish magazine and newspaper publisher said. The transaction called for Mr. Murdoch's News International PLC, a unit of Australia - based News Corp., to subscribe to a rights issue by Zeta valued at 6.65 billion pesetas ($ 57 million). Also participating in the issue was Servifilm Spain Cinematografica S.A. The film producer , owned by Madrid - based financier Jacques Hachuel, received a 5% stake in the Barcelona-based publishing group . The cash injection boosted Zeta 's capital more than four-fold, to 8.47 billion pesetas from 1.82 billion pesetas, greatly enhancing the group's ability to make investments, Zeta officials said. Following its failure last month to win a license for one of Spain's first three private television stations, Zeta is seeking investment opportunities in communications and publishing. With annual sales of about 30 billion pesetas, Zeta publishes over a dozen magazines, including the popular Tiempo, Interviu and Panorama , and three regional dailies . Chairman Antonio Asensio will retain a 70% share in Zeta ." — OntoNotes, `WSJ 1018`, Annotations from OntoNotes

This article, being longer, has much more complicated coreference, so I will look specifically at one chain: Zeta. In this chain, all mentions of "Grupo Zeta S.A." are linked together. However, consider the automatic annotations:

> "Rupert Murdoch acquired a 25% stake in Grupo Zeta S.A., the leading Spanish magazine and newspaper publisher said. The transaction called for Mr. Murdoch's News International PLC, a unit of Australia - based News Corp., to subscribe to a rights issue by Zeta valued at 6.65 billion pesetas ($ 57 million). Also participating in the issue was Servifilm Spain Cinematografica S.A. The film producer , owned by Madrid - based financier Jacques Hachuel, received a 5% stake in the Barcelona-based publishing group. The cash injection boosted Zeta's capital more than four-fold, to 8.47 billion pesetas from 1.82 billion pesetas, greatly enhancing the group's ability to make investments, Zeta officials said. Following its failure last month to win a license for one of Spain's first three private television stations, Zeta is seeking investment opportunities in communications and publishing. With annual sales of about 30 billion pesetas, Zeta publishes over a dozen magazines, including the popular Tiempo, Interviu and Panorama , and three regional dailies . Chairman Antonio Asensio will retain a 70% share in Zeta." — OntoNotes, `WSJ 1018`, Annotations from CoreNLP

We can see that the automatic annotations split the "Zeta chain" into three different coreference chains: one for mentions of the string "Zeta," one for mentions of "the group," and one erroneous chain that connects "its" with "cash injection."

In this case, we lost a set of CAPs from losing `<stake, PREP>` and the "group" chain. However, it's possible we could have gained more CAPs too—if the erroneous

"cash injection" chain had been longer, it could have contributed a <boost, SUBJ> verb-dependency pair to the set of CAPs extracted from this document.

In this specific case we almost witnessed an extension of the set of events visited by the schema learning model through an erroneous coreference link. While it did not happen in this specific case, its easy to see how such accidental "reaches" could contribute to an insignificant improvement in performance with respect to an evaluation metric that relies on identifying events in text blind of any other factors.

All-in-all, we can see in both cases that the models are given nearly identical sets of events within the documents, albeit with some variation. Since the MDL models are event-centric, and the gold standard model has not, in one form or another, predicted more events contained in the held-out documents than the automatic annotations did, then the gold standard measure is not going to produce an improvement with respect to the MDL $|C|_M$ measure.

Thus, at its core, we can conclude that an event-centric measure is, with respect to any notion of schema accuracy, an insufficient measure of schema quality alone. This is surprising given the event-dominant precedent in the literature; the cloze task too is an event-centric measure. However, we also see the relationship between a schema and its ability to either predict or encode events without directly considering the actors involved in holdout data is an implicit one.

This explains the quantitative results seen. However, I would also like to get a qualitative idea of the performance of the schemas. That is, when a schema is applied to a document, does the sense in which it was used in the training data seem to match with the sense it was used in the hold-out data? In the next section, I will take a qualitative look at how well the schemas map the sense of the events used in the training data with the sense used in the hold-out material.

4.9 A Qualitative Inspection of Retention of Sense From Training to Holdout Data

While the measures utilized above yield insignificant differences between schemas generated with the gold standard and automatic annotations, a close inspection of a document, the schemas applied to that document, and the training data used to create the schemas applied can help illuminate qualitatively what "an insignificant difference" actually means. In other words, is any actual knowledge being transferred or generalized from training data to holdout data? Are the matches of schemas that occur simply shotgunning through coincidental event alignments, or do they appear to be successfully aligning situations from source to target material? Are the quantitative measures telling us that the schema matches are equally bad or equally good?

In this section, I look closely at one specific document in the holdout data—`CHTB 0269`, a write-up from a press briefing—with two goals in mind: to obtain a deeper understanding of the effect of the differences in the gold standard vs automatic annotations on the process outlined here, if any, and to see how well the schemas generalized content from the training data to the holdout data.

From close inspection of the MDL scoring process, I obtained the schemas actually applied to obtain MDL scores. The schemas homogeneously applied in these cases are illustrated below.

All said and done, for this specific document, the schemas generated from both gold standard and automatic annotations only matched only two unique event types in the document. The gold standard did better in this specific case, matching three events—*establish* and two instances of *form*—with a single schema (Figure 4.15) for `CHTB 0269`:

> "...the Hong Kong Special Administrative Region 's first legislative assembly and regional organizations will be *formed* according to the Basic Law and relevant resolutions of the National People 's Congress ." — OntoNotes, `CHTB 0269`

> "Currently , the Preparatory Work Committee of the Preparatory Committee of the Hong Kong Special Administrative Region is studying relevant specific issues on *forming* the Hong Kong Special Administrative Region 's first legislative assembly , and will submit suggestions to the Preparatory Committee that will be *established* in 1996." — OntoNotes, `CHTB 0269`

The automatic annotations produced the best-matching schema in Figure (4.16), matching the events *provide* and *violate* in `CHTB 0269`:

> "It is an insult to China 's family planning policy , has seriously *violated* the mission and principles of the UN charter..." — OntoNotes, `CHTB 0269`

> "...we resolutely oppose any country *providing* donations with attached conditions to the population fund..." — OntoNotes, `CHTB 0269`

Note that each individual selection made here is done using the schema presence measure (Section 4.2), which is free from influence of coreference information—aside from that residually implicit from schema generation—as it is unreliable and incompatible between the gold standard and automatic pipelines.

If we trace back the schema generated by the gold standard data, eight documents are returned with chains that contributed CAPs to statistics that contributed to the growth of the schema of interest. One document in particular made a significant

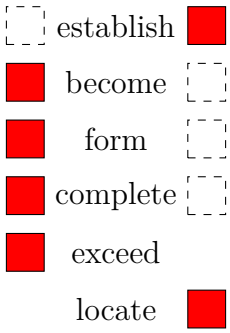


Figure 4.15: Schema generated from gold standard annotations, applied to `CHTB 0269`. The sole argument chain scored highly for the sole role type `district`.

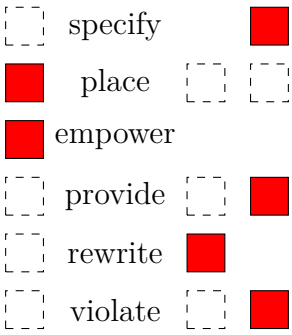


Figure 4.16: Schema generated from automatic annotations, applied to `CHTB 0269`. The sole role chain fell to the fall-back type, `THINGY`. In the source document identified, this seems to be `Article II of the Constitution`.

contribution, with five out of six events attested in a single coreference chain in `CHTB 0161`:

> "The Tianjin Harbor Bonded Area *completed* and *exceeded* all the economic targets for the first quarter of this year...
>
> "The Tianjin Harbor Bonded Area was *established* through State Department approval in May 1991 , in November of the same year , formally invited businessmen , and closed customs and operating in April 1992 . After nearly 5 years of construction , it has *become* the largest and functionally complete bonded area in northern China...
>
> "it has a planned surface area of 7 square kilometers , already developed land of 3.8 square kilometers , accumulated investments in infrastructure facilities of more than 1.2 billion RMB , and has *formed* complete water , electricity , gas , heat , telecommunications infrastructures , etc..."
>
> —OntoNotes, `CHTB 0161`

Two relatively short documents contributed to *locate*'s collocation with *establish*, adding *locate* to the schema. Notably, this represents a similar situation to that described in the hold-out document—the establishment of a special region within China with specific economic rules. The sense of the word *form* is different between the holdout and training document: *form* in the source document is used in an odd way, referring to the development of infrastructure in the special economic district, whereas in the hold-out document, *form* refers to the formation of the district. Nevertheless, the word has two senses that work in both cases.

As for the automatic annotation, similarly, a single chain in a single document—`WSJ 0112` provided the sole contribution to the generating the utilized schema. In this case, it was a single chain linking instances of "Article II of the Constitution:"

"It signals Congress 's attempt , under the pretext of guarding the public purse , to deny the president the funding necessary to execute certain of his duties and prerogatives *specified* in Article II of the Constitution...

"The 1990 appropriations legislation attempts to strip the president of his powers to make certain appointments as *provided* by Article II..."

"Article II *places* on the president the duty to nominate, "and by and with the Advice and Consent of the Senate" appoint , ambassadors , judges , and other officers of the U.S. . It also *empowers* the president to make recess appointments...

"Such laws *violate* the provision in Article II that requires the president to make recommendations to Congress..."

"If President Bush fails to do so in his first year , he will invite Congress , for the remainder of his presidency , to *rewrite* Article II of the Constitution to suit its purposes ."

–OntoNotes, `WSJ 0112`

Again, the schema refers to a similar situation as described in the original article. Where the source article discusses violations of Article II of the Constitution, the hold-out document describes violating the "mission and principles of the UN Charter." Like with the gold standard schema, the second event type to which the schema was applied differs from the source material, in this case, the sense of *provide* is quite different here—*provide* is used to mean give a resource rather than to allot powers.

Nevertheless, it seems in both cases the system used information from a specific and prior document instance to match a fragment of each hold-out document that reflects a remarkably similar situation from the source data, but in both cases did so

with some aberration and error that is interpretable because of the underspecification of senses in the schemas themselves.

Given the small number of documents used to seed the schemas that were applied, this suggests that the schemas learned here are highly unstable—that is, if the core documents that allowed for their growth were removed, the schemas would cease to exist. However, this corpus was also relatively small, so any given document in the training data actually represents a substantial portion of the training corpus. Overall, this suggests that an exploration of the stability of schemas is crucial in understanding how to apply them to the analysis of narratives in text. This stability issue will be explored further on the NYT corpus in Chapter (6).

4.10 Conclusions

In this chapter, I evaluated schemas generated in Chapter (3) with respect to two new measures. The first, the NASTEA measure, looks at how well a single schema or set of schemas can identify a set of salient entities mentioned in a New York Times article. The second, the MDL measures, uses schemas to help encode the narrative content of an article, juxtaposing this encoding with the size of the model used to reach this size. Both of these measures rely on the notion of the *presence* of a narrative schema in a document, which attempts to fit the events in a narrative schema to the document.

Two experiments—one on the New York Times corpus and one on OntoNotes—were conducted using these three germinators.

The New York Times experiments showed the counter-training and random walker schemas outperforming those produced by linear induction and the baseline germinators on both the MDL measures and the N_6 measure.

The OntoNotes experiments failed to show any improvement to schema-based models when they are provided with gold standard data; however, they likely are not appropriate models for the information contained in the OntoNotes data, which generally lacks narratives. A close inspection revealed that the schemas applied successfully generalized situations from the training documents to the holdout ones, but the schemas themselves relied on a single source document for the majority of instances explored.

Of course, these experiments show diversity in data can affect experimental outcomes. The new germinators outperform on the NYT but underperform on the OntoNotes data. However, the evaluation measures used here did not reflect a dramatic change in performance despite dramatically different schemas from different germinators. I will press further to see if differences can emerge under different conditions. Meaningful divisions within a data set ought to have an effect as well. The next chapter will address this, looking at the interactions between narrative schemas, document categories, and topic to see how this additional information can influence the schemas generated for better or worse.

Chapter 5

Narrative Schemas, Document Categories, and Topic Models

5.1 Introduction

At a recent DC NLP meet-up, I gave a talk on some of the research described in this chapter. The audience is largely composed of people from industry positions, and many of them are just curious about what natural language processing is and want to learn more about it. After the talk, one of the audience members asked, "I don't understand, what's the difference between schemas and topics?"

In this chapter, that is the very question I seek to answer. Intuitively, many schemas appear to reflect specific topics or document categories. A schema containing the events "shoot" and "kill" presumably might align with `Crime and Criminals` document category of the New York Times corpus. Given that the gold standard document categories in the NYT corpus are easily accessible, it is not substantially more complex to integrate these categories into the schema framework.

I will outline the content of this chapter after first laying out the results this component of the study might yield.

5.1.1 Possible Relationships

There are atomically two possible entailment relationships between narrative schemas and document categories:

- *schema* $\rightarrow$ *categories*

- *category* $\rightarrow$ *schemas*

In other words, the first hypothesis is that the presence of particular schemas in a document entails that the document is in particular document categories. The second is that a document category entails some set of narrative schemas that are predictably contained within the documents of that category. These questions are largely a matter of degree, as such relationships are largely empirical, and might be related to variable extents.

A note on notation: within the context of these small theorems in this chapter, a slash through the operator will be used to express the absence of a relationship, not that the inverse, converse, or any specific relationship is true. For example, *schema* $\not\rightarrow$ *categories* does not mean $\neg$(*schemas* $\rightarrow$ *categories*), but simply that such a relationship does not exist.

5.1.2 Chapter Contents

In this chapter, I will first address the *schema* $\rightarrow$ *categories* hypothesis in Section (5.2). To this end, I will experiment with different strategies using a Naïve Bayes classifier with features constrained to those that can be inferred from a set of schemas. The evidence in this experiment disfavors the *schema* $\rightarrow$ *categories* hypothesis, though its consequences are open to interpretation. In the second experiment (Section 5.3), I test the converse—*category* $\rightarrow$ *schemas*—and discover variance in the distribution of narrative schemas between document categories and that this variance leads to improved performance on the NASTEA and narrative cloze metrics using models of narrative that are conditioned on document category.

Of course, not in all circumstances are well-defined document categories available for a set of documents. Particularly for documents drawn from the web, there may not be a carefully set of hand-annotated categories available to use for schema

generation. Topic models provide one commonly used, unsupervised alternative to document category annotations. In Section (5.4), I address the use of a topic model as a substitute for document categories. The results do not show that topic models implemented using the strategy devised here do not act as a reliable substitute for document category annotations, though preliminary results indicate that topics could be leveraged in a similar way to the siloing strategy used for document categories.

5.2 Schemas as Predictors of Document Category

In this experiment, I will test whether schemas can be used to predict document category. To this end, given the dataset described below (Section 5.2.1), I generate schemas and explore using different decompositions of them as features for a Naïve Bayes classifier (Section 5.2.2). I describe the results of this process in Section (5.2.4)

5.2.1 Data Selection

The data for this experiment came entirely from the New York Times Corpus (Sandhaus, 2008a), which consists of around 1.8 million documents from the eponymous newspaper. As noted earlier, each document comes tagged with associated metadata, including date, two types of document categories, tags of people mentioned in each document, and other information. In this section, I describe how this corpus was winnowed down for this experiment.

Keyword and Year Selection

All documents containing the keyword "police" in any form were extracted from the New York Times Corpus. Documents from late 1994 to mid 2008 were retained. This reduced the set to roughly 8% of the original corpus size.

Categorical Selection

Documents in the NYT corpus are tagged with an `online_producer` property that provides categorical labels for documents. A subset of these categories was retained, with the intention of providing not only a variety of narratives, but also some more potentially complex distinctions that could be difficult to disentangle. Collectively, this represents a set of documents that are more likely to refer to police as the focus—"noise" and "demonstrations and riots"—than many of those excluded—"international relations" and "United States Armament and Defense." No categories outside of this set were explicitly excluded, however, and nothing prevents these categories from overlapping, which they often do. Most extreme in this regard is the category "Serial Murders", where every article is also contained in "Murders and Attempted Murders."

In total, 38832 documents remain in the corpus of source data. Table (5.1) lists the categories and gives a breakdown of the distribution of documents across categories.

Coreference and Dependency Preparation

Documents were parsed and their coreference chains were extracted with Stanford CoreNLP version 3.4.1 (Manning et al., 2014), particularly the Stanford Parser (de Marneffe et al., 2006) and the Stanford Deterministic Coreference Resolution System (Lee et al., 2013). From the parser, I used the `collapsed-ccprocessed-dependencies`. I only looked at dependencies related to the verb, and each dependency was collapsed into an appropriate super-category: `agent`, `subj`, `nsubj`, `csubj`, `xsubj` are all mapped to `SUBJ`; `comp`, `obj`, `dobj`, `nsubjpass` to `OBJ`; `iobj` and `prep_.*` to `PREP`.[1]

[1]Chambers and Jurafsky (2009) include *prep* as one of their argument slots but do not include it in their diagrams: "An *event slot* is a tuple of an event and a particular argument slot (grammatical relation), represented as a pair $\langle v, d \rangle$ where v is a verb and $d \in \{subject, object, prep\}$."

Schema Generation

For this experiment, I used Chambers' PMI score and the counter-training algorithm (see Section 2.7.1).

5.2.2 Preparing Schemas for Classification Experiments

To better understand the properties of schemas, I investigate how well schemas correlate with the document categories assigned within the NYT corpus. I look at the schemas in two different ways—first, by assigning document categories to schemas, then by using these assignments to complete a categorization task. I do not expect the system to perform better than proven categorization techniques—rather, the categorization task acts as a proxy for investigating the distributional properties of schemas.

Retrieving Category Counts for Schemas

To employ schemas for classification, I interpret them as a set of features. Effectively, if the different event argument slots are nodes of a graph, the chains can be thought of as edges between nodes. These edges are pairs of verb-dependency pairs which I will refer to as *co-referring argument pairs*, a concept introduced in Section (2.7.1).

Aside from the argument types, CAPs preserve all of the information contained inside the schemas themselves, but their separability allows for partial matches and for them to be more easily deployed in the Naïve Bayes classifier. One can think of these as bigrams predicted to exist in a document based on the schemas generated. However, instead of them being derived from collocations of tokens in text, they predict collocations through coreference and specific dependency links—a higher-level, linguistically informed notion of collocation. Schemas represent an aggregation and sampling of these collocations.

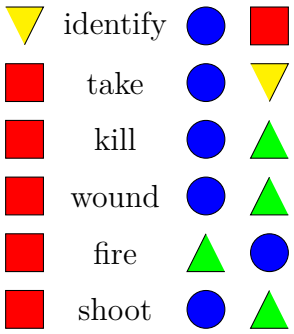


Figure 5.1: A schema extracted using the counter-training technique, generated for and used in the classification task. The red square and blue circle both indicate different PERSONs. The downward pointing yellow triangle indicates some THINGY; the upward pointing green triangle indicates either baghdad or a THINGY. This is the same schema presented in Figure (3.1), but reprinted here for reference.

For example, Figure (5.1) contains a number of different chains. Some CAPs derived from this schema are $\{\langle \text{kill}, \texttt{SUBJ}\rangle, \langle \text{shoot}, \texttt{SUBJ}\rangle\}$ from the red square PERSON chain—derived, intuitively, from the fact that someone who shoots often kills—$\{\langle \text{fire}, \texttt{PREP}\rangle, \langle \text{shoot}, \texttt{OBJ}\rangle\}$ from the blue circle PERSON chain—derived from the fact that one may "shoot someone," but also "fire *at* someone"—among many, many others. This schema alone contains 37 CAPs: 15 each from the two chains that are shared in each and every role slot, and 7 from the other two auxiliary slots.

While types factor into schema extraction, I do not use them explicitly here. Chambers and Jurafsky (2009) demonstrated that using types improves performance on the cloze task over non-typed chains; however, they did not use types on the cloze task itself. Nevertheless, merely using types in schema generation was enough to improve performance. Similarly, while I used types to improve schema generation, I do not use them explicitly in these experiments.

For a given set of chains S^C from schema S, I disentangle the CAPs contained via the following:

$$\text{CAPs}(S) = \{\{vd_a, vd_b\} : \bigwedge_{x \in \{a,b\}} vd_x \in C \in S^C\} \tag{5.1}$$

where C is a chain contained in the set of chains S^C, and vd_x is any verb-dependency pair; a and b are arbitrary indices. I then can assign weights to a category c for a schema S by counting the categories of the documents that each CAP appears in, or more specifically:

$$W(c, S) = \sum_{d \in D} \begin{cases} w(c) : & d \cap \text{CAPs}(S) \neq \emptyset \\ 0 : & otherwise \end{cases} \tag{5.2}$$

where D is the set of sets of CAPs from each of the training documents. $w(c)$ is a weighting function for a category. If working with simple document counts, $w_1(c) = 1$ is sufficient; alternatively, a cf-idf—like tf-idf but with categories instead of terms—could be used. This measure uses $w_{idf}(c) = \frac{N}{n_c}$, where N is the total number of documents in the corpus and n_c is the number of documents denoted as class c.

5.2.3 Classification Experiments

In order to understand the extent to which schematic information interacts with document categories, I considered individual, plausible components of schemas as baselines to compare against the performance of the full blown schema-based CAP classifier. I discuss these in this section, as well as how the classification was performed and how the target data set was chosen.

Each experiment represents a different way of extracting features from each schema. In other words, the schemas are the same between experiments, but the technique for interpreting them changes. Each technique is intended as a plausible

candidate for explaining how the schematic classifier works, working from the simplest to more complex collocations. These techniques include: the "Bag of Words" Model, the Document Co-presence Model, the Coreference Co-presence Model, the Schematic CAP Model. Each will be described in detail below, along with how these different models were implemented.

Experimental Models

In this section, I discuss each of the baseline models, leading up to the features discussed in Section (5.2.2).

Bag of Words Model

The bag of words model used here relies only on the presence of events found in schemas for classification. Instead of thinking of each schema as a set of chains that are decomposed into CAPs, I look at each schema as a set of events S^E:

$$\mathbb{W}(S) = \{v_x : v_x \in S^E\} \tag{5.3}$$

where v_x is a verb and x is an arbitrary integer. The $\mathbb{W}$ of the schema in Figure (5.1) is $\{\texttt{shoot}, \texttt{fire}, \texttt{wound}, \texttt{kill}, \texttt{take}, \texttt{identify}\}$.

Document Co-presence Model

In the document co-presence baseline model, if two events both appear in a document—regardless of their location or presence in a coreference chain—then that counts as an instance of that feature.

$$\mathbb{D}(S) = \{\{v_a, v_b\} : \bigwedge_{x \in \{a,b\}} v_x \in S^E\} \tag{5.4}$$

All permutations of pairs of events are considered. In a schema of size 6, this means that there are 15 pairs of events as features: $\{\{\texttt{shoot}, \texttt{fire}\}, \{\texttt{shoot}, \texttt{wound}\}, \ldots \textit{ etc.}\}$.

Coreference Co-presence Model

The final baseline creates pairs of any two events which share co-referrent arguments. I do not include the specific argument slot. Now using S^C, the set of chains from schema S, instead of S^E:

$$\mathbb{C}(S) = \{\{v_a, v_b\} : \bigwedge_{x \in \{a,b\}} v_x \in S^C\} \tag{5.5}$$

This model's features are nearly schematic in nature, except that the features lack the specific slot wherein co-presence was defined; at this point, I effectively am using schemas without their role slot labels. Features derived from the schema in Figure (5.1) are no different from the last baseline because all events are shared with at least one chain. However, the interpretation of the hold-out documents changes. Because I am now looking at coreference, it is not the mere presence of a pair of events in the text, but their linkage through their arguments via coreference that counts.

Schematic CAP Classifier

This is the schematic classifier, as discussed above and illustrated with Equation (5.1). Note that Equation (5.5) is nearly identical to Equation (5.1); v has been swapped with vd representing the set of verb-dependency pairs. With verb-dependency pairs instead of verbs alone, I have constructed a set of features that closely approximates each set of schemas.

Implementation

I used the `scikit-learn` class `sklearn.naive_bayes.MultinomialNB` to classify the documents (Pedregosa et al., 2011). Because the document categories overlap, I took a one-vs-all classification strategy for each document class; each document category represents a split into `+` or `-` classes. For the classification task, to give as much information as possible to the classifier, I generated 800 schemas seeded with the 800 most frequent verbs. I held-out $1/10^{th}$ of documents for evaluation.

In performing classification, I conducted a "rank descent." I started with the highest weighted category for a given feature in the first test. These categories were applied to their respective documents, then checked against the gold standard for that test. This process was then repeated, using the two highest-weighted categories instead of just the first in the second experiment, etc., until every category that appeared with the feature is applied. These ranks are represented as n in Table (5.1).

I completed the classification task in two separate sets of experiments using the raw counts weighting (w_1) in one and the cf-idf (w_{idf}) weighting scheme in the other.

5.2.4 Results

Table (5.1) contains a breakdown by category of peak performance. Categories that were better represented tend to have higher peak F1 scores. More poorly represented categories tended to peak in performance with the CAPs or at least coreference information provided by the coreference co-presence model $\mathbb{C}$, though this was not entirely the case—the very frequent category "crime" peaked with the $\mathbb{C}$.

Figures (5.2) and (5.3) illustrate precision-recall curves for both series of *up to rank n* experiments. In all cases, n goes up moving from left to right; recall increases with each increase in n.

Table 5.1: Number of documents per category retained from the "police" subset with classification performance. Particularly, this includes the rank n at which the rank descent reached the peak F1 value, which of the weighting functions w_x—w_1 or w_{idf}—was used from Section (5.2.2) and which of the models was used from Section (5.2.3) for which performance peaked with respect to F1. $\mathbb{W}$ is the bag of words model, $\mathbb{D}$ is document co-presence, $\mathbb{C}$ is coreference co-presence, and CAPs represents a fully schematic classifier. N is the number of documents in a respective category. Some category names have been shortened or abbreviated.

Category	N	F1	n	w_x	Model
Terrorism	16,290	0.422	9	*idf*	$\mathbb{W}$
Crime	14,685	0.461	6	*idf*	$\mathbb{C}$
Murders	13,872	0.430	1	1	$\mathbb{W}$
World Trade Ctr.	8,916	0.213	3	1	CAPs
Violence	6,450	0.183	5	*idf*	$\mathbb{C}$
Demonstr. and Riots	6,430	0.193	4	*idf*	$\mathbb{W}$
Accidents	5,719	0.166	4	1	$\mathbb{W}$
Police Brutality	4,627	0.237	2	1	$\mathbb{W}$
Blacks	3,522	0.166	6	*idf*	$\mathbb{D}$
Law and Legislation	3,319	0.321	2	1	$\mathbb{W}$
Frauds	1,848	0.136	7	*idf*	$\mathbb{D}$
Attacks on Police	1,621	0.168	3	1	$\mathbb{C}$
Organized Crime	871	0.098	4	*idf*	$\mathbb{C}$
Serial Murders	918	0.075	8	1	CAPs
Cocaine	464	0.061	5	*idf*	CAPs
Suburbs	303	0.108	3	*idf*	CAPs
Noise	206	0.037	14	1	$\mathbb{D}$
Prison Escapes	137	0.100	2	*idf*	CAPs

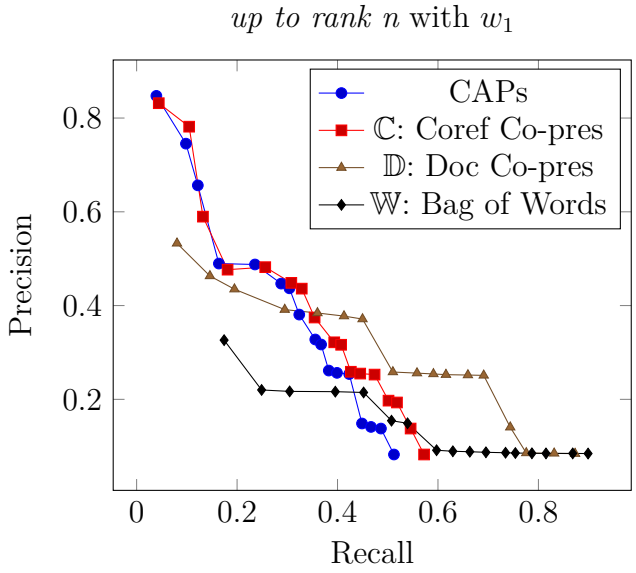


Figure 5.2: Precision/Recall curves for the *up to rank n* classification experiment using w_1 to assign categories to schemas.

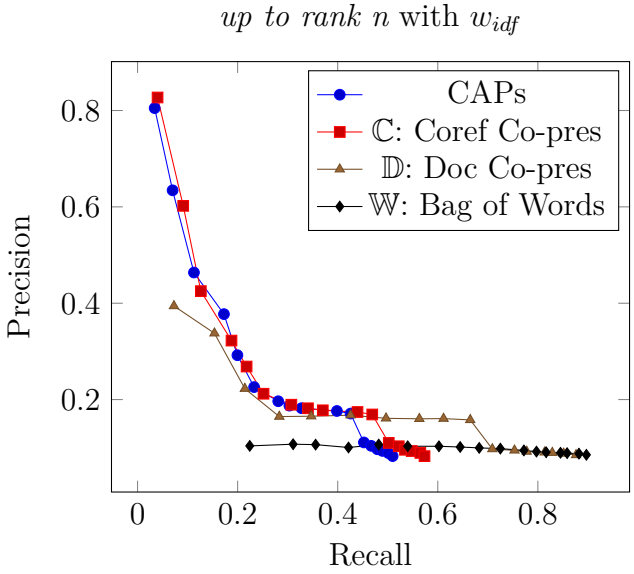


Figure 5.3: Precision/Recall curves for the *up to rank n* classification experiment using w_{idf} to attach category assignments to schemas.

5.2.5 Discussion

Remarkably, there is some capability for schema-specific features to classify documents despite being generated without any explicit knowledge of the classifications they denote. Not in all cases is this the best, but it tends to help bolster performance in under-represented categories within the corpus. The precision-recall curves in Figures (5.2) and (5.3) illustrate this point—as features are removed that the schemas uniquely provide, the peak precision generally declines. This shows that the features included in schemas do possess information specific to their associated document categories.

Of course, the rather simplified classifiers I've presented are by no means reflective of an industry standard classifier.[2] The number of features—only 6,901 unique CAPs available, 1,629 word types in the $\mathbb{W}$ baseline—is less than what would be available to a typical bag of words analysis on the same data set—193,702 word types. This performance produces precision-recall curves with a concave shape. However, what I do see is a suitable illustration that, with respect to the relationship between schemas and categories, the whole is greater than the sum of its parts.

Also worth noting is the fact that the precision-recall curve of the schematic classifier and the coreference co-presence classifier $\mathbb{C}$ nearly adhere to one another. Figure (5.1) gives a great example of why slot information may not be helpful in all circumstances. In this schema, there are two very clear individuals in most of the events: a shooter of some sort, and someone who was shot. What about with *identify* and *take*? These are a bit more ambiguous; the precise utility of each exact argument slot is not as clear. The connections created through coreference, however, are extremely

[2]While F1 scores across categories averaged 0.199, a non-schematic, bag-of-words Naïve Bayes classifier using all available word types averaged 0.458. Most categories outperformed the non-schematic classifier, except for Suburbs and Prison Escapes, which scored 0.000 with the non-schematic classifier.

precise, albeit losing recall. This puts into question approaches that leave out coreference (Balasubramanian et al., 2013; Cheung et al., 2013)—with respect to this task, something was lost without it.

It is also necessary to critically question the precision of the source annotations, especially the largely unknown criteria used by the NYT Indexing Service to determine document categories. With respect to the schema in Figure (5.1), most individuals indubitably would say that such a schema is associated with murder. However, there are plenty of examples where *shooting*, *wounding*, and *killing* are not classified by the NYT Indexing Service as "Murders and Attempted Murders:"

> "A Brooklyn grand jury has cleared two police officers in the killing of an unarmed man whom they shot 18 times..."
>
> "The United States Marshal who shot and wounded a Queens high school student Thursday after mistaking the candy bar he was holding for a revolver..."
>
> "...the Police Department is being scrutinized over the shooting of several civilians by officers... a Hispanic teen-ager was shot in the back last month in Washington Heights."

In the words of Joe Strummer, "murder is a crime, unless it is done by a policeman" (Strummer, 1982). While I did not apply the types of role fillers explicitly to the classification task, these sorts of "errors" motivate the use of role fillers in future work.

5.2.6 Conclusions

I have shown techniques for deriving features from narrative schemas, and shown that features derived from narrative schemas are more than the sum of their parts.

In particular, coreference information is a crucial component of them and seems to produce the most substantial boost in precision of any of the features of schemas used.

Additionally, the schemas themselves are not remarkably good predictors of document category, in part because the content of a particular schema can spread across multiple document categories. While they help somewhat, they're not particularly exceptional classifiers of documents. In other words, there are two conclusions to draw from this:

- $schema \not\rightarrow category$: a narrative schema does not predict the category of a specific document category.
- $schema \not\leftrightarrow category$: schemas are not equivalent to document categories.

This begs the question about a converse relationship that hasn't been disproven yet: $category \rightarrow schema(s)$, i.e. whether document category predicts a particular narrative schema or set of narrative schemas. I explore this in the next section.

5.3 Some Document Categories are Narratologically Homogeneous

As described in the previous section, there is at best a weak relationship as to whether the presence of schemas predict the category of a particular document. In this section, I explore the converse relationship—whether better schemas can be generated if their narrative models are conditioned on document category.

In Section (5.3.1), I describe the data set used for this experiment. In Section (5.3.2), I describe the techniques applied for schema generation. In Section (5.3.3), I describe how cloze was deployed in this experiment. In Section (5.3.4), I describe how the NASTEA task was deployed for this experiment. In Sections (5.3.5) and (5.3.6), I describe the results and discuss them.

5.3.1 Data

The data for this experiment comes, again, from the New York Times corpus (Sandhaus, 2008a), a corpus containing 1.8 million articles from the New York Times from January 1987 to June 2007. For this experiment, I chose to work on a different dataset, the `op1-sem` dataset discussed in Chapter (3). To reiterate, I select a subset of document categories in the corpus that appear with a similar frequency and represent a broad range of topics (Table 5.2). The schemas used in this study are induced from this set of documents. In one procedure, the entire set of documents serves as the corpus for a single set of schemas. In a second, I create a topic-specific set of schemas, using the set of documents assigned to a given topic as the corpus for a set of schemas. One aim of this is to investigate the extent to which evaluation measures are affected by topic specificity. A second is to examine how the sets of topic-specific schemas might differ.

Table 5.2: Counts of document categories selected from the `online producer` tag for use in this study. Frequencies vary, but were chosen to be around the same order of magnitude and to represent different sorts of topics.

`online producer` category	counts
Law and Legislation	52110
Weddings and Engagements	51195
Crime and Criminals	50981
United States Armament and Defense	50642
Computers and the Internet	49413
Labor	46321
Top/News/Obituaries	36360

Once the documents of these categories were extracted, they were pre-processed using Stanford CoreNLP (Manning et al., 2014). Of particular importance are the Stanford Parser (de Marneffe et al., 2006) and `dcoref` (Lee et al., 2013), used for

coreference resolution. These play a central role in the schema generation process described in the next section. Documents where parsing or coreference failed to complete were removed from processing as well.

5.3.2 Modifications to Schema Generation

While the counter-training technique produces interesting schemas, it is relatively slow, considering all the possible scores that must be considered against each other. Instead, I use Chambers and Jurafsky (2009)'s original generation technique, with some modifications.

The model employed here departs fundamentally from Chambers and Jurafsky (2009)'s in that it is conditioned by document category, in this case selected from the `online producer` categories from the NYT corpus that I was interested in. Separate models are trained for each document category, only on documents contained in that category. The only exception to this is the baseline model, which is trained on all documents into one single model. Plausibly, the resulting schemas should be "more topic-specific" than those generated by the baseline model, which lumps all topics together.

Conditioning schema generation by document category, as noted above, is one key difference. Additionally, there are a few small changes at some of the post-score steps in the procedure. The score value from Chambers and Jurafsky (2009) does not explicitly describe how the various slots from an event newly added to a schema should be connected into forming chains within that schema. This occurs in a separate step—after it is decided that an event should be added to a schema, connections are made at that point where the threshold can be crossed. Also, an event may be added to multiple schemas if the score is high enough. In part, this allows for the word's meaning to be captured across multiple contexts.

Lastly, I genericize some types—similar to Balasubramanian et al. (2013)—but not in all circumstances; instead, I do so only in the event that there is no common noun available to learn from. The algorithm first checks the Stanford NER (Finkel et al., 2005) to see if there are any available types. Then it checks if there are any pronouns in the chain, and attempts to guess a type for the chain based on that. Finally, if there are no other types available, it aborts to a fallback type.

During the process of generation, a random selection of 10% of documents were held out for evaluation.

Figure (5.4) depicts a schema generated by this procedure.

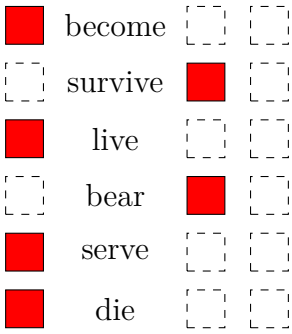


Figure 5.4: A relatively simple schema from the *Top/News/Obituaries* document category. The red squares indicate a chain that is strongly represented by the generic type PERSON, but with many other lionizing human types: scholar, hero, advocate, philosopher, etc. The dashed squares represent slots attested in the data but not connected during schema generation.

5.3.3 Narrative Cloze

Recall that narrative cloze evaluates accuracy on a "fill in the blank" task. In each document, every coreference chain long enough to be considered in the model is considered. One verb is removed from that coreference chain, and the model produces scores for every possible replacement. The rank of the true replacement in these scores

acts as the score for that chain. The model itself is scored based on the average of these chain-wise scores.

To evaluate the *topical* model, separate models are trained on each document category considered. The relevant category is applied based on which category a document is a member of. In parallel, a *flat* model was prepared with the same training data, but by lumping all documents into one single model.

The original cloze task Chambers & Jurafsky (2008) assigned a score of the lowest possible rank +1 to verbs that were not contained in the model. If this were done with the topical model, however, it would rank higher by definition, since words that were missed in a specific topic model would be ranked much higher than in the flat model. To account for this, I assign the lowest rank +1 of the flat model to words missed in the topical model.

Since documents contained in multiple categories of interest are rare, these documents were ignored. They represent a small fraction of the data, and the decision of which topical model to apply in these cases is non-trivial, as some decision has to be made as to whether the document or chain is more appropriate for one topic or the other.

In the deployment of the cloze task used here, I use only a 324-document subset of the hold-out documents. These were randomly selected, one from each bin of 324 separate bins of holdout data. Only a portion of the documents can be clozed, as the procedure takes some time.

5.3.4 NASTEA in this Experiment

In addition to the cloze task, I used the NASTEA task to evaluate the schemas produced (Section 4.3).

I split the data by document category, then generated schemas for each category. In evaluation, only schemas generated with documents from a specific category were applied to that specific category. Analogously, this was done for the narrative cloze task, but instead of schemas, each model—learned from the documents in that one single category—was applied to predict events for that specific category. In both experiments, documents that were members of multiple categories, about 9% of the held-out 27498 documents, were removed from the hold-out data to remove any possible penalties due to categorical overlap. Because executing the task takes some time, the test itself was ran on a random sample of the held-out material totaling 324 documents.

5.3.5 Results

Table 5.3: Average rank of answers in the narrative cloze.

Test Model	Avg. Rank
Baseline	1329
Topical	1273
Top/News/Obituaries	565
Weddings and Engagements	1058
Law and Legislation	1279
Labor	1297
Crime and Criminals	1268
Computers and the Internet	1346
United States Armament and Defense	1805

Of the narrative schemas generated,[3] around 13% of the schemas generated were shared between document categories on average. Each categorical set of schemas shares around 26% of its schemas with the baseline set.

[3]The schemas are available for download at `http://schemas.thedansimonson.com`.

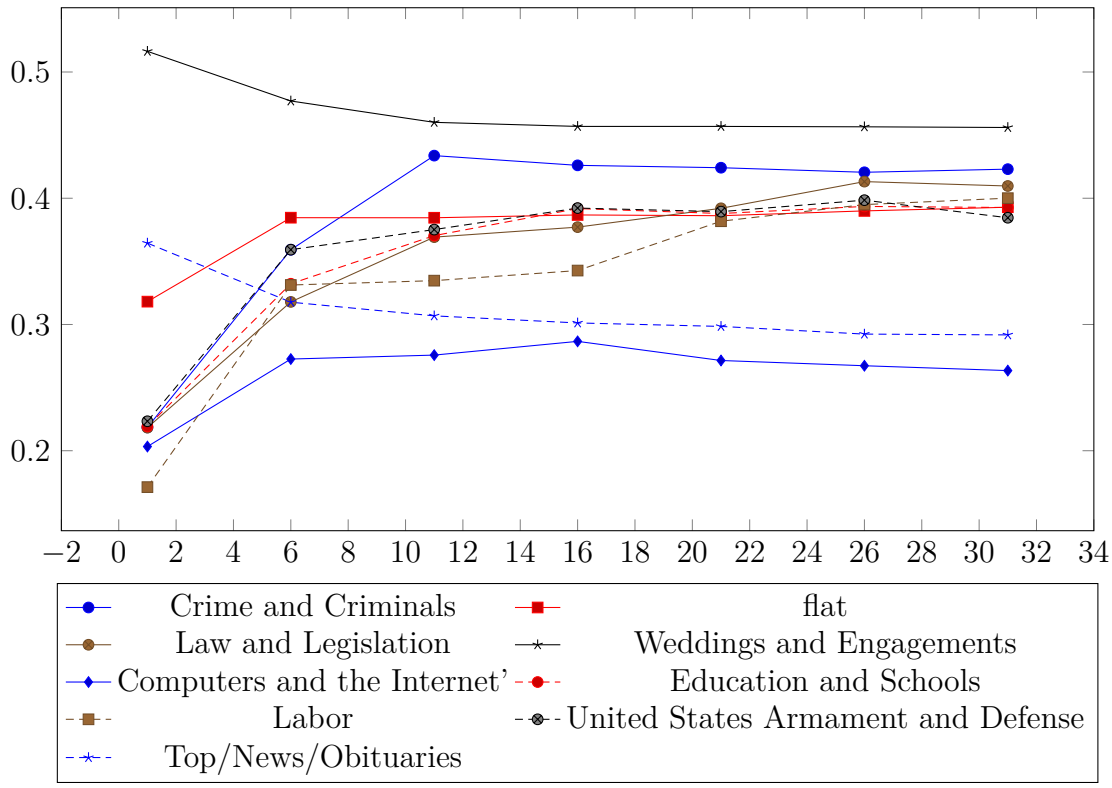


Figure 5.5: Plot of test-by-test performance on the NASTEA task for each topic. The x-axis indicates number of top-n present schemas applied. The y-axis indicates F1 score (i.e. N_n).

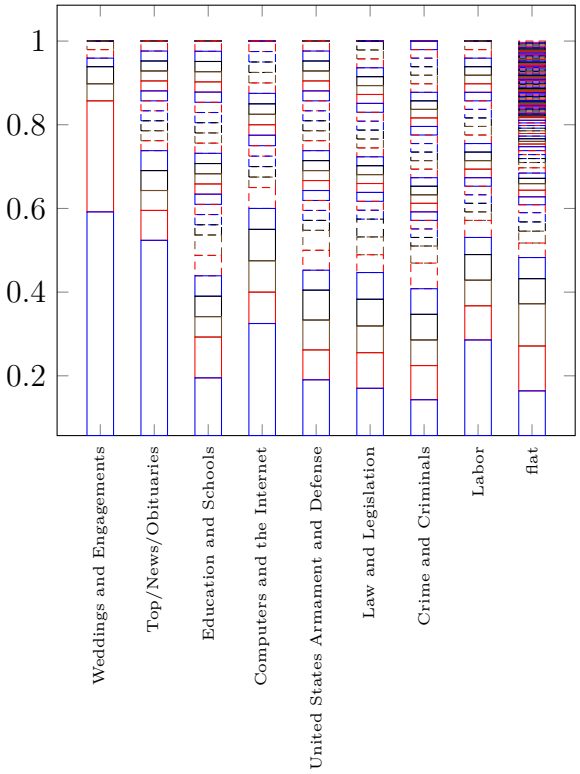


Figure 5.6: Plot of N_1 Document Categorical Narrative Homogeneity. A representation of the variety of schemas with the highest presence across all documents in a category ($n = 1$ for the NASTEA task). Fewer slices of the whole represent a smaller fraction of schemas being applied scoring as ever having the highest presence in that topic. A larger slice indicates that the single schema it represents had the highest presence for more documents in that topic than a smaller slice.

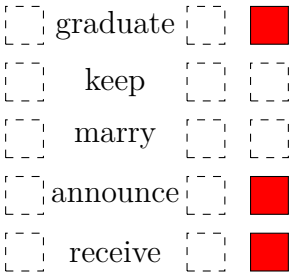


Figure 5.7: Schema generated in the *Weddings* document category. The dashed squares represent slots attested in the data but not connected during schema generation. The chain of red squares indicates a generic organization type.

Table (5.3) contains the cloze task results. Figure (5.5) illustrates results for the NASTEA task, broken down by document category. Most categories follow a general trend of performing poorly with the highest-presence guess alone. As more schemas are applied, the system is better able to retrieve annotated entities on most categories, with F1-scores leveling off around 45%. These values remain more or less stable *ad infinitum* with a few minor variations in value as n continues to increase. The "flat" baseline model follows this trend as well.

However, two categories are exceptions to this trend: *Weddings and Engagements* and *Top/News/Obituaries*.

This exceptional N_1 performance necessitates closer inspection. Since NASTEA is applying schemas to documents, those schemas can be retained and counted allowing for illustration of the variety of different schemas that seem to best fit a particular document, what I will refer to as *narrative homogeneity*. Figure (5.6) takes the N_1 results and illustrates the totals of counts for schemas that were applied in each N_1 case. Categories that performed well on N_1 were also more homogenous at N_1, choosing a single schema as most present more often than their more heterogeneous counterparts.

5.3.6 Discussion

The NASTEA task shows a clear, discrete distinction between two types of document categories: those that seem to be narratologically homogeneous and others that seem to be narratologically heterogeneous within the scope of this model of narrative. In the homogeneous case, the assertion that $category \rightarrow schema$ seems to be valid, while in more heterogeneous circumstances, this is much less the case. This affirms Miller et. al. (2015)'s observation that their own corpus is characterized by a "heterogeneity of the articles' foci," with their corpus likely fitting into the *United States Armament*

and Defense category—a notably heterogeneous one—were it derived from the NYT Corpus.

Of those used in this study, the *Weddings and Engagements* and *Top/News/Obituaries* (referred to hereafter as *Weddings* and *Obituaries*, respectively) are distinctly homogenous. This is reaffirmed through the cloze task as well, where each of their respective rank averages are hundreds of ranks higher. This indicates that they are more rigid in their choice of wording and the events they describe, and those events point more strictly toward the entities the NYT library scientists annotated. It is not too surprising that these particular categories are different. Impressionistically, such documents have writing styles that are more formulaic than their more news-typical counterparts. However, the objective measurability of this impression is a first.

There are two possible interpretations of this result. One is that the homogenous categories are truly something different from the heterogeneous ones, and that this is a fact about news narratives and document categories at large. The other interpretation is that the homogenous categories are ones that are better encapsulated by this model of narratives and that the heterogeneous ones are not captured properly. These are not necessarily contradictory interpretations, if one accepts both of them as independently representing different interpretations of the notion of narrative.

While cloze and NASTEA agreed overall on the exceptionality of *Weddings* and *Obituaries*, there remain some discrepancies between the two. *Obituaries* performs much better on cloze relative to *Weddings*, while on NASTEA, the reverse happens, and *Weddings* outperforms *Obituaries*. Within the rest of the categories, rankings shuffle around between the two. For example, *Computers and the Internet* performed well below average on cloze, but ranked third highest on N_1, with the homogeneity to match. Narrative cloze's opacity makes these discrepancies difficult to understand

without trolling through thousands of rankings. NASTEA has the transparency to show what is going on under the hood: clear differences in narrative homogeneity.

These results are wonderful, but if generalizing these techniques to new corpora without document category annotations, can a topic model be used in lieu of such annotations? I experiment with using a topic model in schema generation in the next section.

5.4 Schemas and Topics

In Section (5.3), I have shown that document category can sometimes have an influence on the content of narrative schemas. Specifically, this association has been shown between schemas generated from the New York Times corpus and the document categories tagged therein by its library scientists. If I apply the techniques used here to a new data set—e.g. one gathered from the web—I would not have a hand-annotated gold standard set of document categories to influence the new models with.

Thus, given the strong influence of document categories on schemas, it would behoove any unsupervised analysis using narrative schemas to supply some sort of substitute for them. The readily and most accessible substitute for document categories would be a *topic model* of the provided corpus. Topic models are a set of unsupervised algorithms that model documents through a generative model. Such models assume that documents underlyingly are a mixture of topics and that the bag of words that constitutes a document is a result of this mixture of topics.

I chose topic models generated via Latent Dirichlet Allocation (LDA, Blei et al. (2003)) to represent topics in this dissertation. This choice is largely due to their popularity; LDA is often used as a go-to topic model (Özbal et al., 2016; Sim et al., 2016; Tran & Ostendorf, 2016; Wilson et al., 2016) in many CSS applications. Often topic

models alone are used to explore corpora in an unsupervised manner (Wallach, 2011). While I will not show, as initially hoped, how schemas could potentially enhance such studies, it's hoped that this demonstration of the interaction of schemas and topics will come closer to closing that gap.

5.4.1 Document Categories vs. Topic Models

The terms *document category* and *topic* are often used interchangeably. However, there are a number of foundational differences between the two that prevent their direct and obvious interchange. In this section, I want to juxtapose the meaning of document category and topic model to make crystal clear what makes the problems in this chapter different from those in Sections (5.2 & 5.3).

Table 5.4: Comparison between document categories and topic models.

document category	*topic models*
Supervised: annotated gold standard categories	Unsupervised: generated without training data
Underlyingly Intensional	Extensional
Membership is discrete and absolute	Membership is fractional and probabilistic
`op1sem`: Selected to be independent	Not guaranteed to be independent

First, document category, as the term is used in this dissertation, refers to the hand-annotated categories in the NYT corpus. This is not necessarily always the case, as a document classifier can be trained to assign document categories automatically. While derived from hand-annotated tags, document classification can be thought of as originating with human-assigned and defined categories. Topic models, on the other hand, do not originate with human-assigned categories or defined categories. They merely arise from the data provided. This doesn't mean they couldn't be a assigned

by hand in theory, but they don't necessarily arrive with a pre-packaged definition that entails or excludes explicitly any specific propositions as membership criteria. Since topics lack explicit definition, they are open to interpretation.

Second, document categories are intensional while topic models are extensional. This relates back to my first point—document categories are tagged on to documents because of the human assigned definitions of those categories. This definition is the intension of the document category. Topics, on the other hand, are derived from examples of text. They are born as divisions among distributions of tokens, and they are often represented as distributions of tokens. This makes them extensional in nature, existing fundamentally as examples of instances of things in the topic, and although an intension may be implied, they remain fundamentally represented by examples of things contained each topic, making them extensional.

Third, document categories are absolute, discrete, and Boolean in their membership. A document either is in a category or is not. Topics, however, are probabilistic and fractional. A document is either in the `Obituaries` category or it is not, but a document might be 50% one topic i and 50% topic j. The topic distribution for a document is a set of probabilities, so their sum must equal one. Similarly, a document may wholly be contained in multiple discrete document categories, but a document is partially distributed over topics, and those distributions do not intersect. A document might be both in the `Labor` and `Obituaries` document categories, but no document could be both 66% in some topic i and 66% in some topic j.

Fourth, the document categories in the `op1sem` corpus were selected to be generally independent of one another, and while topics are intended to be independent of one another—something Latent Dirichlet Allocation prefers—they are not guaranteed to be independent of one another. This is only a condition of the categories used in this subcorpus, and often document categories are not independent of one another.

In Table (5.4), I juxtapose these differences directly.

The differences juxtaposed above have practical concerns. The approach taken previously in Sections (5.2 & 5.3) I retroactively refer to as *siloing*—that is, separate and independent models are created for each document category. All of the documents for training models for each document category were separated, and from each "silo," separate sets of schemas were generated. This leveraged the discrete and absolute nature of document categories, which defined what was inside a silo and what was not. Document categories absolute definition and interpretability contributed to schemas' utility since every schema could be interpreted within the context of a well-defined categorical scope; that is, a schema containing "bear," "survive," "lived," "graduate," "work," etc. is much more clearly interpretable by humans when it is considered a recurring member of the `Obituaries` document category. Additionally, topics are not guaranteed to be independent—a great deal of documents will have membership spread across multiple topics. Some of these will be minor contributions but non-zero nevertheless. These contributions must be handled properly.

However, it should be clear that the siloing used to document categories does not map well onto topic models. Thus, I implement in this chapter a simple modification of the scoring used in previous chapters. This modification uses $chainsim'$ as previously deployed but weights $chainsim'$ by the similarity between the distribution of topic weights associated with a candidate event—i.e., a verb—and the corresponding distributions of the events already contained in a schema being generated. Thus, this new score generates a different sort of schemas, *topical schemas*, as opposed to the previously generated *categorical schemas* in previous chapters. The procedure for generating these will be discussed and evaluated in this chapter.

5.4.2 Integrating Topic Models into Schema Generation

To make this integration as simple as possible, I implement a substitute for $chainsim'$ that is identical in arguments and type returned. Thus, I define a topical form of $chainsim'$ as $chainsim_\tau$:

$$chainsim_\tau(S, v) = topicsim_\tau(S, v) \times chainsim'(S, v) \tag{5.6}$$

Thus, $chainsim_\tau$ can seamlessly substitute $chainsim'$ in any germination process. However, this leaves much of the complexity to *topicsim*. This I define as,

$$topicsim_\tau(S, v) = \tau(\Psi(S)) \cdot \tau(\psi(v)) \tag{5.7}$$

τ is a function that maps a sequence of tokens to a normalized vector whose weights are proportional to the topic weights assigned by the topic model. More specifically, τ uses gensim's `getdocumenttopics` to obtain topic weights, then treats them as a vector and divides each of the topic weights by the magnitude of the whole vector.

Because schemas are not documents, nor are the candidate verbs, there are an infinite number of possible ways to interpret them in a way that fits into a topic model. The specific implementation here—Ψ and ψ—I refer to as the *pseudodoc* model, described in the next section.

5.4.3 Pseudodocs

Because gensim's `getdocumenttopics` requires a sequence of tokens, I force schemas and candidates into this mold for interpretation by the topic model.

In the candidate case, the single verb is wrapped up as a token in a singleton sequence, so ψ can simply be described as:

$$\psi(v) = \langle v \rangle \tag{5.8}$$

However, Ψ is a bit more complicated. The intuition here is that the distribution of tokens in the pseudodoc should be somewhat like the distribution of tokens that inspired the schema in the first place. Every chain has multiple possible types that each chain represents, and each chain has events that are linked through those chains. Thus, one way to go about representing the schema as a sequence of tokens is to traverse the possible types of each chain, and for each event linked in the chain, add a pair of each event verb and type to the pseudodoc.

This intuition is captured in Algorithm (5): First, the Ψ attempts to generate a

Algorithm 5: Pseudodoc algorithm for converting a schema into a sequence of tokens for the purpose of giving a schema topic assignments.

```
Data: a schema
Result: pdoc, a sequence of tokens
events_raw, chains = schema
pdoc = []
for events, types in chains do
    for type in types do
        for verb, dep in events do
            pdoc.extend([type, verb]);
    if not types then
        for verb, dep in events do
            pdoc.extend([verb]);
if not pdoc then
    pdoc = [e for e,D in events_raw]
return pdoc;
```

pseudodoc from the chains of the schema. If types are available to traverse, then the pseudodoc is pairs of each combination of event verb and chain type, with each event verb only being those linked to the specific chain being considered. The ordering here is irrelevant since the tokens are converted to a bag of words before being given to the topic model. The types are those which resulted in the highest score during previous

iterations. If no types are available for a given chain—particularly the case when the schema is initially seeded—then Ψ returns the event verbs linked to slots in the chain. If for whatever reason the above generates an empty pseudodoc, then Ψ returns a list of event verbs contained in the schema as a last resort.

Thus, these definitions for Ψ and ψ allow us to generate pseudodocs to represent narrative schemas and their candidate inductions in a topic model.

5.4.4 Topic Models Generated

In this section, I describe briefly the precise procedure used to generate topics.

To start, I pre-processed the documents with CoreNLP. From each document, I extracted the lemmas of each token, excluding words that were in `gensim`'s stopwords list.[4] From these lists of lemmas, hapax legomena were removed.

I used `gensim`'s LDA model to create a topic model of these lists of lemmas.[5] Most settings were kept at their default values. The number of passes over the data was set to 20.

I generated four different topic models with each one varying by number of topics: 8, 24, 64, and 100 topics. 8 was chosen as the smallest to correlate with the number of document categories in the source material. 24 represents a hypothetical configuration of 3 topics : 1 document category; 64 represents 8:1; and so forth. There is no guarantee that these configurations actually play out this way; however, this ideal inspired these choices of numbers of topics.

I tried two variants of input tokens to the topic models. One was based on all of the tokens in each document; the other was based solely on the entity types contained

[4] `gensim.parsing.preprocessing.STOPWORDS`

[5] `https://radimrehurek.com/gensim/`

in slots that were used in chains to generate schemas. These are denoted with an "E-" in front of the number of topics used.

5.4.5 Data

In this experiment, I again use the `op1-sem` subcorpus of the NYT corpus (Sandhaus, 2008a). However, in this case, I use the training / dev split from Section (4.7). For generation, I did not do full parameter climb, but I did use the dev set to determine the number of schemas used n to score the test set.

Testing at multiple parameter values in this experiment is the equivalent of adding another step on to the parameter climb executed previously, though admittedly potentially with the same issues—particularly, that the parameters explored first are not necessarily independent of those set last which were arbitrarily set, thereby setting the parameters first tested to values that are optimal for the default values of those last tested.

5.4.6 Results

No specific modifications were made to the NASTEA task in this case, similar to how Chambers & Jurafsky (2009)—which added slot types to the model it was creating—used the cloze task without overt modifications to include types to the task.

5.4.7 Discussion

As shown in Tables (5.5) and (5.6), using topic model information does not appear to improve schemas, at least with respect to the NASTEA task, in neither the whole text topic models or the entity driven ones. With respect to Counter-training, performance degraded, and with respect to the Random Walker, performance improved in one case (8 topics), but this could not have been ascertained from the results on the dev set.

Table 5.5: NASTEA task F1 scores for schemas generated with topical influence. $|T| = 1$ refers to the original parameter climb from Chapter (4.7.1).

	$\|T\|$	Dev N_n	Test N_n	n	λ	β	c	s	S
CT	1	0.416	0.415	22	1.0	2.0	3	5	100
CT	8	0.408	0.412	46	1.0	2.0	3	5	100
CT	24	0.412	0.412	41	1.0	2.0	3	5	100
CT	64	0.411	0.413	34	1.0	2.0	3	5	100
CT	100	0.389	0.409	31	1.0	2.0	3	5	100
RW	1	0.422	0.407	8	0.5	1.0	5	12	100
RW	8	0.412	0.417	26	0.5	1.0	5	12	100
RW	24	0.413	0.404	35	0.5	1.0	5	12	100
RW	64	0.408	0.396	22	0.5	1.0	5	12	100
RW	100	0.404	0.407	11	0.5	1.0	5	12	100

Table 5.6: NASTEA task F1 scores for schemas generated with topical influence. This includes topics generated from only entity mentions used to inform slot types in schema generation. $|T| = 1$ refers to the original parameter climb from Chapter (4.7.1).

	$\|T\|$	Dev N_n	Test N_n	n	λ	β	c	s	S
CT	E-1	0.416	0.415	22	1.0	2.0	3	5	100
CT	E-8	0.407	0.422	37	1.0	2.0	3	5	100
CT	E-24	0.409	0.411	49	1.0	2.0	3	5	100
CT	E-64	0.408	0.416	40	1.0	2.0	3	5	100
CT	E-100	0.333	0.345	22	1.0	2.0	3	5	100
RW	E-1	0.422	0.407	8	0.5	1.0	5	12	100
RW	E-8	0.403	0.399	13	0.5	1.0	5	12	100
RW	E-24	0.408	0.414	15	0.5	1.0	5	12	100
RW	E-64	0.407	0.413	29	0.5	1.0	5	12	100
RW	E-100	0.340	0.335	17	0.5	1.0	5	12	100

The dev scores did not predict very well the best performers on test. This may indicate that a more thorough parameter climb may be helpful for generating better schemas with a topic model. In this case, I simply adopted the parameters that performed best training on "one topic." Perhaps starting with a few more would have given a set of parameters that better utilized the information contained in the topic model once it came time to optimize for the number of topics parameter.

It is worth noting that other means of injecting a topic model into schema generation—either through a more formal process or in a way that is more deeply integrated into scoring the chains—could indeed deliver improvements in the schemas generated with respect to the NASTEA task. However, it's clear from these results that a simple approach of just strapping a topic model to a event and chain associations model through some arbitrary connection of positivity is unlikely to yield much improvement.

These results further emphasize the contributions of document category to the schema generation process. The distinction between heterogeneous and homogeneous categories demonstrated by the NASTEA task—also reflected in improvements in scores overall from the NASTEA task—are not seen here. It seems that topic models do not exactly behave in the same way that document categories do and are not necessarily one-to-one replacements.

Of course, it is possible that with the right parameter settings on the LDA topic model, a set of topics could result that do better reflect the document categories desired. These could also change the narrative here drastically. However, in the short term, there is no evidence of such a set of parameters existing, and even if they did, there is no guarantee that they would translate to successfully extracting similarly distinct categories from a new corpus. Obtaining similarly useful results on a new

data set may require new parameters, and a new sort of experiment for identifying the point at which the topic model can successfully draw these desired distinctions.

This requires closer inspection. In Figures (5.8, 5.9, 5.10, and 5.11), we can see that members of some document categories largely fit into a single or handful of LDA topics, especially the homogenenous categories, and especially when the number of topics is closer to the number of document categories. In other words, while all documents are a mixture of topics, one topic often has a majority stake of the probability, and if not a majority, often a clear plurality. As the number of topics increases, the document categories become more diffuse, spreading into more topics. Sometimes, these topics intersect across document categories, though not often. This suggests that further experiments could be conducted where LDA topic assignments are used to silo documents, like was done with document categories. In particular, such an experiment—via varying the number of topics—could provide an interesting test for a "good" number of topics for a corpus.

Notably, the topic model failed to distinguish the homogeneous document categories from one another until the number of topics reached 100, while schemas are extremely sensitive to the differences between these two document categories under most observed parameter conditions. This suggests still that an approach that properly blends the two types of models could provide a powerful technique for an unsupervised analysis.

5.4.8 Conclusions

This section implemented a schema score that integrated a topic model to score candidate event verbs for schemas with respect to both the narratological fitness and the topical fitness of the candidate event. This was intended to draw similar distinctions to those found in document categories in Section (5.3), since siloing schema models

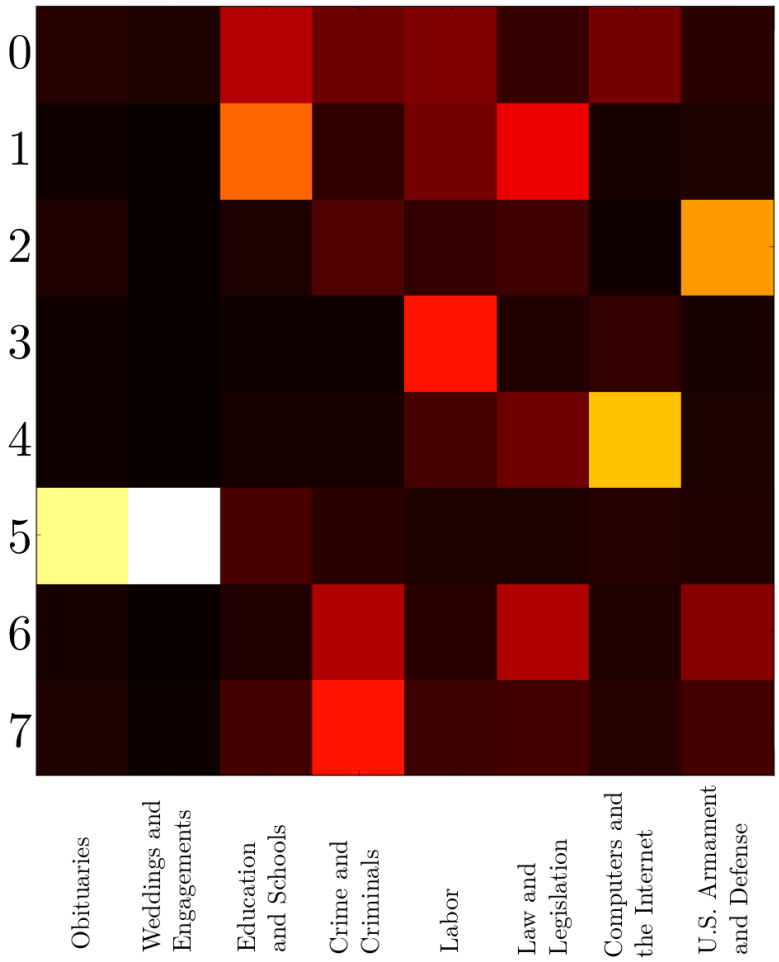


Figure 5.8: Heat maps of the LDA topics (8) that the documents in each `op1-sem` document category fall into.

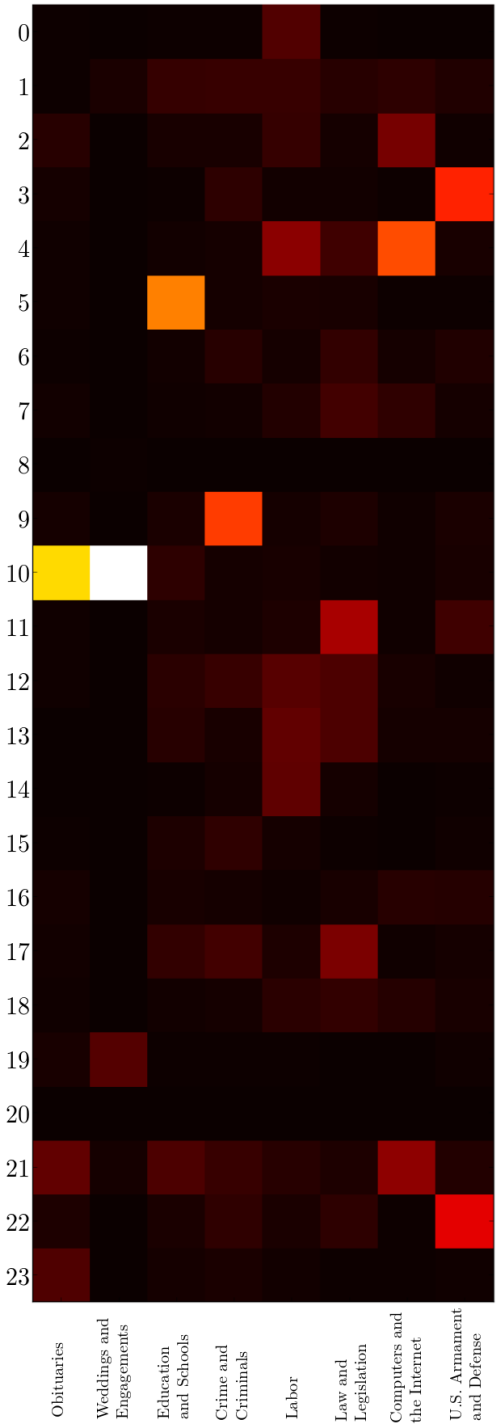


Figure 5.9: Heat maps of the LDA topics (24) that the documents in each `op1-sem` document category fall into.

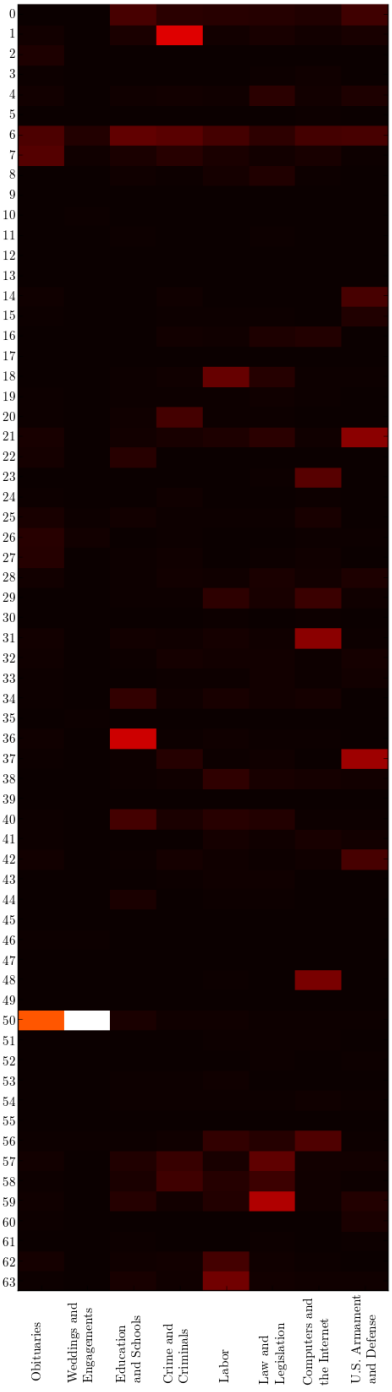


Figure 5.10: Heat maps of the LDA topics (64) that the documents in each `op1-sem` document category fall into.

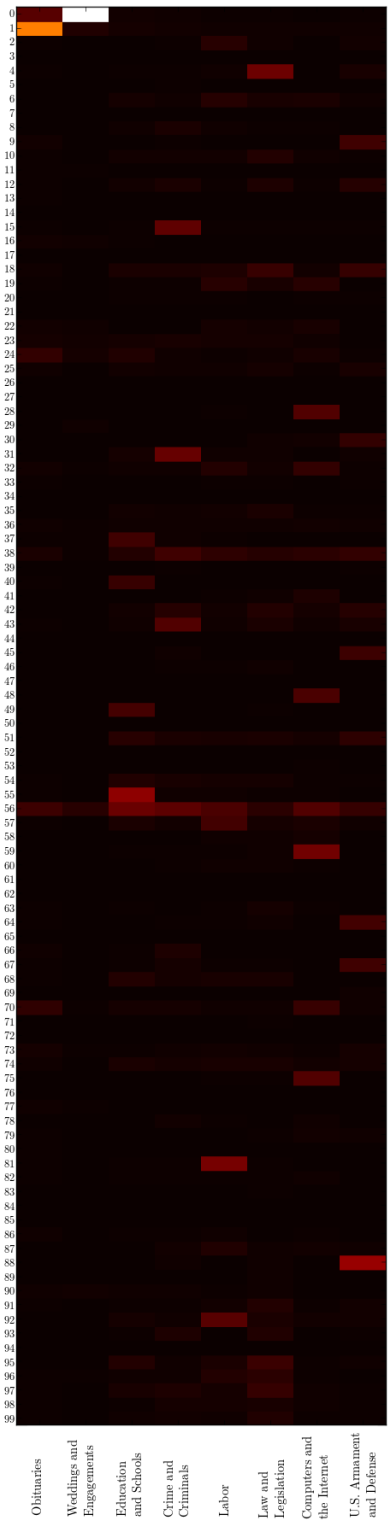


Figure 5.11: Heat maps of the LDA topics (100) that the documents in each `op1-sem` document category fall into.

by category contributes to improved performance. This integration treated the topic distribution of a candidate event as a vector, and used what I coined as the *schema pseudodoc* to find an appropriate topical fit for each schema. The dot product between the candidate vector and the schema's vector was used to weight the scores.

However, scores in evaluation largely did not show improvements. While it may still be possible to find a set of parameters that better contribute to schema quality, it seems that a simple implementation of a topic model into a schema model does not result in improvements. This also suggests that topic models are not a simple one-to-one substitution for document categories, especially with respect to the qualities of document categories that led to improved schema generation previously.

In the next section, I will conclude this chapter and consider the consequences of the results contained therein.

5.5 Conclusions

The results of both experiments paint a relatively clear picture of how narrative schemas relate to document categories: that there is a strong, one way relationship, where document categories may predict the presence of particular schemas, but the converse relationship is tenuous at best.

I will state affirmatively that $category \rightarrow schemas$ if the category is homogenous. For the other theorems, the evidence currently points to them not being valid in all circumstances, and the circumstances under which they are valid are unclear.

It seems that these results for document categories do not immediately or easily generalize to topics from an LDA topic model. There are, of course, many ways to iterate upon the approach taken here. Carefully searching the set of parameters for LDA better or choosing a entirely different algorithm for generating topic could

potentially result in topics that better reflect the document categories used in Section (5.3). However, it is worth keeping in mind that the homogeneous / heterogeneous distinction dropped out of the results with little coaxing. It seems to reflect a natural distinction between the document categories that is robust from many directions of analysis. Because the experiments using information from topic models fail to reflect this distinction, their ability to do so remains insubstantiated.

There are a few possible follow-ups here, but for now, they are outside of the scope of this study. None of the known homogenous document categories were attested in the *schemas* → *categories* experiments—repeating this experiment with the homogenous categories could reveal stronger predictive capabilities for those categories. It is possible that many other homogeneous categories exist. The two found here were by chance, but it is possible that others exist. A thorough exploration of the entire NYT corpus' document categories could uncover others.

In addition to considering the document category or categories of a corpus to be analyzed, it is also important to understand how much data must be collected to draw conclusions from it. In the next chapter, I address this through a number of experiments centered around schema stability.

Chapter 6

Schema Stability

6.1 Introduction

As the last chapter showed, the document categorical context of a text can have a strong influence on the homogeneity of the schemas present within it. Similarly, in understanding how schemas can be applied to analyze the narrative content of text, we need to understand how fluctuations in the sample of text obtained can affect the schemas generated from it.

Just as word tokens, schemas should persist under repeated, variable observation to provide maximal utility. Any word type worthy of analysis should persist across texts. If a few texts are removed at random from a corpus, it should not affect the general statistics of the word types, given that those word types are attested robustly. This is the rationale for removing low-count word tokens from most statistical analyses. Similarly, any set of narrative schemas should be robust enough to persist given variations in the data set.

Similarly, when conducting an unsupervised study with (or of) narrative schemas, we need to know when enough documents have been collected to make reliable inferences about the data or when too few documents are available to make such inferences. In other words, at what point have enough documents been collected that the schemas yielded from them gives a complete picture of the narratives contained from the collection source?

To establish the stability of schemas, I have used a procedure where documents are removed at random from the training data used to generate schemas. Specifically, I have carried out a series of ablation and cross-validation studies, reducing the corpus from which schemas are generated by up to 90%. If the schemas are *stable*, then a few documents removed should not dramatically alter the content of the schemas themselves to a tremendous degree. Broadly, the results reflect general stability, and as measured here increasing stability corpus becomes quite reduced. There are some complications involved in evaluating the similarity between two sets of schemas—as opposed to the similarity between two individual schemas—which I discuss in the following section.

6.2 Data and Procedure

This analysis is performed on the `op1-sem` data set, first introduced in previous chapters. These documents were selected for membership in one the categories included in Table (5.2), which themselves were chosen for describing generally independent phenomena in the world and for being of a similar, manageable order-of-magnitude.

Table 6.1: An exact facsimile of Table (5.2). Counts of document categories selected from the `online_producer` tag for use in this study. Frequencies vary, but were chosen to be around the same order of magnitude and to represent different sorts of topics.

`online producer` category	counts
Law and Legislation	52110
Weddings and Engagements	51195
Crime and Criminals	50981
United States Armament and Defense	50642
Computers and the Internet	49413
Labor	46321
Top/News/Obituaries	36360

For NLP preprocessing, I used the Stanford CoreNLP suite of tools (Manning et al., 2014),[1] as was used in the rest of the dissertation. CoreNLP was chosen for having both parsing and coreference facilities (de Marneffe et al., 2006; Lee et al., 2013), both of which are essential for generating schemas. The full pipeline includes pre-requisites for those tasks, including but not limited to tokenization and part of speech tagging (Toutanova et al., 2003; Toutanova & Manning, 2000), both of which are referenced in performing higher-level information extraction tasks. The named entity annotations provided by the CoreNLP Pipeline are still leveraged as well(Finkel et al., 2005).

The stability procedure contains two dimensions: an ablation and a cross-validation step. The first dimension is the ablation. Between ablations, 10% of the *total* set of documents is removed—not 10% of the last ablation.

The second dimension is the cross-validation. Here, within each cross-validation, the documents are split ten ways, then $9/10^{ths}$ of the available documents are used to generate schemas. These splits are not preserved across ablations.

While the procedure was performed by removing parts from the whole, the most intuitive way to interpret the meaning of this procedure is in reverse—that is, to think of some sort of search procedure resulting in slightly different results (cross-validation step) at each step in a larger data collection (ablation step).

To spell this out explicitly, consider a case where we have 1,000,000 documents. Ablation 0 will have all 1,000,000 documents. Each cross-validation in Ablation 0 will contain 900,000 documents. Ablation 1 will have 900,000 of the original 1,000,000 documents. Each cross-validation in Ablation 1 will contain 810,000 documents. Ablation 2 will have 800,000 documents, and its cross-validations will contain 720,000 documents, and so forth.

[1]Stanford CoreNLP, Version 3.4.1 (2014-08-27)

Given what I determined in the last chapter, schemas should be generated separately by document category. This further compounds the complexity here, as each ablation and cross-validation is done with respect to each document category. More homogeneous document categories are far more likely to be more stable than heterogeneous ones. Therefore, all the schemas generated in the experiments described in this chapter are category-specific; that is, they are generated solely from documents assigned to one of the NYT corpus categories in Table (6.1).

Document categorical distinctions are retained as they were in Chapter (5) as well when generating schemas.

From each ablation/cross-validation pair, I prepared separate PMI-based models and created separate sets of schemas for each of the germinators discussed in Chapter (3). To understand how similar these sets of schemas are, I will use the Fuzzy Jaccard coefficient defined in Section (3.9.1). However, there are some issues with respect to Fuzzy Jaccard's interpretability, which I will address in the next section.

6.2.1 Interpreting Fuzzy Jaccard Values with the Jaccard Reciprocal Fraction

In Section (3.9.1), I defined the "Fuzzy Jaccard" measure J_{J_e} to measure similarity between sets whose members may be partially similar. To reiterate:

$$J_{J_e}(S,T) = \frac{|S \cap_{J_e} T|}{|S| + |T| - |S \cap_{J_e} T|} \tag{6.1}$$

This is essentially the original Jaccard co-efficient with the denominator substituted with an identity. While union has been successfully avoided, intersection itself must be defined. I define it as:

$$|S \cap_{\kappa} T| = \sum_{\tau \in T} \max_{\sigma \in S} \kappa(\sigma, \tau) \tag{6.2}$$

where κ is a similarity measure defined between all elements of S and T and always returns a value between or equal to 0 and 1. $|\sigma \cap_\kappa \tau|$ was directly defined to circumvent the problem of defining an intersection between partially similar elements. Specific to schemas, we define J_e as a similarity measure between two schemas based on the events contained therein:

$$J_e(\sigma, \tau) = \frac{|\sigma_e \cap \tau_e|}{|\sigma_e \cup \tau_e|} \tag{6.3}$$

This measure as a whole was defined to specifically allow partial matches between sets of sets. That said, if J_e were a boolean equivalence relation, then this would reduce to the original Jaccard measure.

While Fuzzy Jaccard gives some notion of similarity between two sets, it is misleading on its own. The properties of the Jaccard coefficient tend to skew things very quickly from 100%, especially when considering schemas. Just at the level of schema-to-schema comparisons, when two schemas σ and τ differ by one event, the resulting J_e value is:

$$J_e(\sigma, \tau) = \frac{|\sigma_e \cap \tau_e|}{|\sigma_e| + |\tau_e| - |\sigma_e \cap \tau_e|} = \frac{5}{6 + 6 - 5} = 71\% \tag{6.4}$$

While the schemas share 5/6 events (83%), the eventive Jaccard J_e only scores at 71%, since the schemas are themselves "punished" for having two events that do not match.

This is not the whole story though. Because the full Fuzzy Jaccard measure J_{J_e} is a formula of similar form, it further exaggerates these small differences. Let's assume all of the schemas in two sets of schemas have a J_e value as described above at 71%. In other words, these are fundamentally two very similar sets of schemas—all schemas contained in each set have a counterpart in the other set that shares 5/6 events. I'll also assume both sets are the same size. This gives us:

$$J_{J_e}(S, T) = \frac{|T| \times 0.71}{2|T| - |T| \times 0.71} = 55\% \tag{6.5}$$

Glancing at such a value is misleading. It implies the sets of schemas are only 55% similar despite each schema in one set sharing 5/6 events for every schema in the other set.

What we really want to know is what the typical similarity between two sets of schemas is—in other words, does each schema typically have a counterpart that shares 3/6 events? 4/6? 5/6? This is really what we want such a measure to tell us. In this section, I will derive such a transformation from the Fuzzy Jaccard value, albeit with a few assumptions about the schemas themselves that generally hold true here.

To start, consider the Fuzzy Jaccard coefficient, as defined in Chapter (3)

$$J_{J_e}(S,T) = \frac{|S \cap_{J_e} T|}{|S| + |T| - |S \cap_{J_e} T|} \tag{6.6}$$

Where $|S \cap_{J_e} T|$ is defined as:

$$|S \cap_{J_e} T| = \sum_{\tau \in T} \max_{\sigma \in S} \frac{|\sigma_e \cap \tau_e|}{|\sigma_e| + |\tau_e| - |\sigma_e \cap \tau_e|} \tag{6.7}$$

First of all, I assume that $|\sigma_e| = |\tau_e|$, given that the termination condition in many of these algorithms is $|\sigma_e| >= \sigma'$. Thus, I'll assume that $|\sigma_e| = |\tau_e| = \sigma'$:

$$|S \cap_{J_e} T| \ = \sum_{\tau \in T} \max_{\sigma \in S} \frac{|\sigma_e \cap \tau_e|}{2\sigma' - |\sigma_e \cap \tau_e|} \tag{6.8}$$

There is really no way to ascertain $\max_{\sigma \in S} |\sigma_e \cap \tau_e|$ except experimentally because this value, the number of events shared between two schemas, is the value we want to solve for. To make this possible, I assume that there exists a typical value x such that $\max_{\sigma \in S} |\sigma_e \cap \tau_e| = x$. I use the word *typical* here since I am not going to prove this is the average here; however, I have a hunch that x is the average value, or at least a related to the average through some linear transformation. The goal from here on out is to solve for this value so that we can transform any J_{J_e} value into a typical value of $|\sigma_e \cap \tau_e|$, henceforth referred to as x. This simplifies the calculation a bit:

$$|S \cap_{J_e} T| \ = \sum_{\tau \in T} \frac{x}{2\sigma' - x} \tag{6.9}$$

Now that the content of the summation is constant with respect to τ, I reduce the summation, giving:

$$|S \cap_{J_e} T| \ = |T| \frac{x}{2\sigma' - x} \tag{6.10}$$

This can then be substituted into the Fuzzy Jaccard formula:

$$J_{J_e}(S,T) \ = \frac{|T| \frac{x}{2\sigma'+x}}{|S| + |T| - |T| \frac{x}{2\sigma'-x}|} = \frac{|T|x}{(2\sigma' - x)(|S| + |T|) - |T|x} \tag{6.11}$$

Since $|S| \ = |T|$, I'll substitute $|S| + |T| = 2|T|$.

$$J_{J_e}(S,T) \ = \frac{|T|x}{(2\sigma' - x)(2|T|) - |T|x} = \frac{x}{2(2\sigma' - x) - x} = \frac{x}{4\sigma' - 3x} \tag{6.12}$$

Now I solve for x, the typical number of shared events between two schemas, in terms of J_{J_e}, giving:

$$x = \frac{4\sigma'}{J_{J_e}^{-1}(S,T) + 3} \tag{6.13}$$

This gives us the typical size of the set of events shared between two sets of schemas based on their Fuzzy Jaccard value. I'd like to point out something interesting about this relationship—σ', the size of the schemas in S and T, only scales the value linearly and can easily be factored out. Removing σ', I define the remaining component as the Jaccard Reciprocal Fraction, or JRF:

$$JRF = \frac{4}{J_{J_e}^{-1}(S,T) + 3} \tag{6.14}$$

This gives us, the fraction of typical shared events between schemas in two sets of schemas, regardless of the size of schemas in each set, only based on the Fuzzy Jaccard value. As the Fuzzy Jaccard value approaches 1, so does the JRF; as the Fuzzy Jaccard value approaches 0, the denominator approaches infinity, and thus the JRF approaches 0. To further put the JRF into context, I state the fact that:

$$x = JRF \times \sigma' \tag{6.15}$$

Or in plain English, the typical number of events shared between schemas in two sets of schemas is equal to the JRF times the typical, equal size of a schema in each set. For the purposes of this study, this is truly the quantity of interest—not the typical number of events themselves, but the fraction of events shared.

In the next section, I will report Fuzzy Jaccard values as the raw computed values of similarity between each set but with the JRF value to aid in their interpretation. Note that the assumptions here hold in many, but not all circumstances. I will discuss, when necessary, how the assumptions applied here may have distorted the JRF values.

6.3 Results

Once I obtained schemas for each ablation, cross-validation, and document category—around 640,000 for those germinators that generate around 800 schemas per category—it is necessary to compare them in some way. For each individual pair of sets of schemas within an ablation, I computed Fuzzy Jaccard scores. I then computed the mean and standard deviations for these scores, and they are reported as such, as well as transformed into JRF form. These values are presented in Tables (6.2 & 6.3). This table remains quite complicated though, so I additionally averaged values across germinators and document categories to produce Figures (6.1 & 6.2).

The Fuzzy Jaccard measure is defined in a way that produces asymmetric results; the Fuzzy Jaccard values in this chapter are only computed in one direction. These were computed only once and in no way optimized. Also, a two-way comparison would complicate any measures of confidence computed, requiring extra caution to avoid giving too much confidence where it is not due.

Category	Abl	Counter-Training			LI-Trunc			Linear Induction			Random Walker		
		μJ_e	s	JRF	μJ_e	s	JRF	μJ_e	s	JRF	μJ_e	s	JRF
Computers and the Internet	0	0.490	0.032	0.794	0.110	0.002	0.332	0.494	0.005	0.796	0.300	0.010	0.631
Computers and the Internet	1	0.481	0.024	0.788	0.110	0.002	0.331	0.489	0.005	0.793	0.300	0.015	0.632
Computers and the Internet	2	0.487	0.021	0.792	0.111	0.002	0.333	0.486	0.005	0.791	0.297	0.011	0.629
Computers and the Internet	3	0.494	0.028	0.796	0.113	0.002	0.338	0.486	0.006	0.791	0.302	0.014	0.634
Computers and the Internet	4	0.499	0.030	0.800	0.115	0.002	0.341	0.481	0.005	0.788	0.297	0.013	0.628
Computers and the Internet	5	0.504	0.026	0.803	0.117	0.002	0.347	0.477	0.004	0.785	0.298	0.011	0.629
Computers and the Internet	6	0.504	0.022	0.803	0.124	0.002	0.361	0.477	0.005	0.785	0.298	0.010	0.630
Computers and the Internet	7	0.515	0.025	0.810	0.132	0.003	0.379	0.469	0.006	0.780	0.314	0.012	0.647
Computers and the Internet	8	0.519	0.029	0.812	0.144	0.003	0.403	0.463	0.006	0.775	0.324	0.015	0.657
Computers and the Internet	9	0.541	0.032	0.825	0.208	0.005	0.513	0.464	0.009	0.776	0.350	0.018	0.683
Crime and Criminals	0	0.485	0.020	0.790	0.106	0.002	0.321	0.485	0.006	0.790	0.334	0.008	0.667
Crime and Criminals	1	0.489	0.018	0.793	0.107	0.002	0.323	0.482	0.005	0.788	0.326	0.009	0.659
Crime and Criminals	2	0.488	0.026	0.792	0.107	0.002	0.325	0.473	0.005	0.782	0.323	0.012	0.656
Crime and Criminals	3	0.481	0.031	0.788	0.107	0.002	0.324	0.471	0.004	0.780	0.317	0.011	0.650
Crime and Criminals	4	0.494	0.036	0.796	0.108	0.002	0.327	0.468	0.004	0.778	0.319	0.012	0.652
Crime and Criminals	5	0.501	0.029	0.801	0.110	0.002	0.330	0.467	0.007	0.778	0.313	0.011	0.645
Crime and Criminals	6	0.506	0.024	0.804	0.112	0.002	0.336	0.463	0.005	0.775	0.318	0.010	0.651
Crime and Criminals	7	0.503	0.025	0.802	0.119	0.002	0.351	0.463	0.005	0.775	0.326	0.010	0.659
Crime and Criminals	8	0.509	0.026	0.806	0.129	0.003	0.373	0.453	0.006	0.768	0.317	0.011	0.649
Crime and Criminals	9	0.531	0.026	0.819	0.161	0.004	0.435	0.441	0.005	0.759	0.327	0.015	0.661
Education and Schools	0	0.490	0.017	0.794	0.110	0.002	0.332	0.496	0.004	0.798	0.311	0.010	0.643
Education and Schools	1	0.498	0.021	0.799	0.112	0.002	0.334	0.494	0.004	0.796	0.316	0.009	0.649
Education and Schools	2	0.500	0.023	0.800	0.113	0.002	0.337	0.495	0.004	0.797	0.317	0.010	0.650
Education and Schools	3	0.503	0.020	0.802	0.114	0.002	0.339	0.492	0.005	0.795	0.320	0.010	0.653
Education and Schools	4	0.504	0.019	0.802	0.116	0.002	0.344	0.485	0.005	0.790	0.324	0.012	0.657
Education and Schools	5	0.492	0.020	0.795	0.117	0.002	0.346	0.482	0.005	0.788	0.321	0.014	0.654
Education and Schools	6	0.510	0.026	0.806	0.121	0.003	0.356	0.473	0.005	0.782	0.319	0.012	0.652
Education and Schools	7	0.514	0.031	0.809	0.128	0.002	0.371	0.472	0.006	0.781	0.329	0.022	0.662
Education and Schools	8	0.522	0.022	0.814	0.146	0.003	0.406	0.460	0.005	0.773	0.335	0.014	0.668
Education and Schools	9	0.532	0.027	0.820	0.199	0.005	0.499	0.444	0.007	0.762	0.353	0.015	0.685
Labor	0	0.477	0.016	0.785	0.112	0.002	0.335	0.485	0.005	0.790	0.314	0.011	0.647
Labor	1	0.478	0.017	0.786	0.112	0.002	0.334	0.484	0.004	0.789	0.316	0.008	0.649
Labor	2	0.480	0.024	0.787	0.114	0.002	0.339	0.480	0.005	0.787	0.317	0.011	0.650
Labor	3	0.488	0.023	0.792	0.115	0.002	0.342	0.481	0.005	0.788	0.318	0.012	0.651
Labor	4	0.490	0.027	0.794	0.118	0.002	0.348	0.480	0.004	0.787	0.311	0.013	0.643
Labor	5	0.493	0.033	0.796	0.121	0.002	0.355	0.476	0.005	0.784	0.309	0.013	0.641
Labor	6	0.511	0.029	0.807	0.124	0.002	0.362	0.472	0.007	0.782	0.314	0.013	0.647
Labor	7	0.518	0.031	0.811	0.134	0.003	0.383	0.476	0.006	0.784	0.320	0.014	0.653
Labor	8	0.529	0.020	0.818	0.151	0.003	0.416	0.464	0.007	0.776	0.337	0.011	0.670
Labor	9	0.546	0.020	0.828	0.207	0.005	0.510	0.449	0.006	0.765	0.345	0.012	0.678

Table 6.2: Schema stability values across four document-categorical experiments.

Category	Abl	Counter-Training			LI-Trunc			Linear Induction			Random Walker		
		μJ_e	s	JRF	μJ_e	s	JRF	μJ_e	s	JRF	μJ_e	s	JRF
Law and Legislation	0	0.462	0.021	0.775	0.114	0.002	0.340	0.511	0.005	0.807	0.318	0.007	0.651
Law and Legislation	1	0.477	0.021	0.785	0.115	0.002	0.341	0.511	0.005	0.807	0.314	0.010	0.647
Law and Legislation	2	0.473	0.025	0.782	0.115	0.002	0.341	0.507	0.006	0.804	0.310	0.008	0.642
Law and Legislation	3	0.481	0.029	0.787	0.118	0.002	0.348	0.504	0.005	0.802	0.309	0.012	0.641
Law and Legislation	4	0.475	0.024	0.783	0.120	0.002	0.352	0.499	0.006	0.799	0.302	0.011	0.634
Law and Legislation	5	0.481	0.020	0.788	0.122	0.002	0.358	0.492	0.005	0.795	0.305	0.008	0.637
Law and Legislation	6	0.493	0.022	0.795	0.128	0.002	0.370	0.490	0.005	0.794	0.307	0.011	0.639
Law and Legislation	7	0.503	0.021	0.802	0.133	0.003	0.381	0.487	0.006	0.791	0.311	0.010	0.644
Law and Legislation	8	0.510	0.027	0.806	0.151	0.003	0.416	0.474	0.007	0.783	0.325	0.012	0.658
Law and Legislation	9	0.528	0.021	0.817	0.201	0.006	0.501	0.461	0.008	0.774	0.353	0.014	0.685
Top/News/Obituaries	0	0.521	0.022	0.813	0.117	0.002	0.347	0.441	0.006	0.759	0.362	0.013	0.694
Top/News/Obituaries	1	0.521	0.018	0.813	0.120	0.002	0.353	0.436	0.004	0.755	0.361	0.013	0.693
Top/News/Obituaries	2	0.516	0.028	0.810	0.121	0.002	0.356	0.432	0.007	0.753	0.368	0.017	0.699
Top/News/Obituaries	3	0.528	0.021	0.817	0.123	0.002	0.359	0.432	0.005	0.753	0.364	0.016	0.696
Top/News/Obituaries	4	0.521	0.027	0.813	0.129	0.003	0.371	0.432	0.008	0.752	0.356	0.012	0.689
Top/News/Obituaries	5	0.532	0.030	0.820	0.135	0.003	0.384	0.422	0.007	0.745	0.351	0.013	0.683
Top/News/Obituaries	6	0.533	0.024	0.820	0.147	0.004	0.408	0.423	0.007	0.746	0.359	0.010	0.692
Top/News/Obituaries	7	0.551	0.034	0.831	0.171	0.006	0.452	0.417	0.008	0.741	0.344	0.016	0.677
Top/News/Obituaries	8	0.554	0.033	0.832	0.212	0.008	0.519	0.412	0.010	0.737	0.368	0.016	0.700
Top/News/Obituaries	9	0.560	0.030	0.836	0.298	0.010	0.629	0.410	0.013	0.735	0.379	0.017	0.710
US Armament and Defense	0	0.461	0.018	0.774	0.108	0.002	0.327	0.500	0.004	0.800	0.319	0.014	0.652
US Armament and Defense	1	0.468	0.016	0.779	0.109	0.002	0.329	0.498	0.004	0.799	0.306	0.007	0.639
US Armament and Defense	2	0.465	0.017	0.777	0.111	0.002	0.332	0.500	0.004	0.800	0.303	0.009	0.635
US Armament and Defense	3	0.471	0.025	0.780	0.111	0.002	0.333	0.497	0.006	0.798	0.309	0.011	0.642
US Armament and Defense	4	0.477	0.022	0.785	0.113	0.002	0.337	0.493	0.004	0.795	0.314	0.008	0.647
US Armament and Defense	5	0.484	0.019	0.790	0.115	0.002	0.341	0.485	0.006	0.790	0.306	0.008	0.639
US Armament and Defense	6	0.489	0.019	0.793	0.118	0.002	0.349	0.485	0.006	0.790	0.309	0.008	0.641
US Armament and Defense	7	0.491	0.026	0.794	0.125	0.003	0.364	0.471	0.008	0.781	0.310	0.014	0.643
US Armament and Defense	8	0.496	0.023	0.797	0.138	0.004	0.391	0.465	0.005	0.777	0.309	0.008	0.641
US Armament and Defense	9	0.517	0.030	0.811	0.180	0.005	0.467	0.453	0.008	0.768	0.332	0.016	0.666
Weddings and Engagements	0	0.555	0.027	0.833	0.279	0.009	0.608	0.471	0.009	0.780	0.412	0.017	0.737
Weddings and Engagements	1	0.560	0.040	0.836	0.295	0.006	0.626	0.474	0.007	0.783	0.415	0.024	0.740
Weddings and Engagements	2	0.559	0.029	0.835	0.310	0.009	0.643	0.467	0.007	0.778	0.401	0.014	0.728
Weddings and Engagements	3	0.558	0.034	0.834	0.326	0.011	0.659	0.467	0.011	0.778	0.412	0.015	0.737
Weddings and Engagements	4	0.560	0.038	0.836	0.338	0.013	0.672	0.466	0.014	0.777	0.407	0.019	0.733
Weddings and Engagements	5	0.568	0.024	0.840	0.351	0.012	0.684	0.456	0.014	0.770	0.420	0.013	0.743
Weddings and Engagements	6	0.578	0.033	0.846	0.385	0.014	0.714	0.465	0.016	0.776	0.435	0.016	0.755
Weddings and Engagements	7	0.591	0.051	0.853	0.416	0.016	0.740	0.458	0.017	0.772	0.440	0.028	0.758
Weddings and Engagements	8	0.616	0.046	0.865	0.459	0.020	0.772	0.459	0.020	0.772	0.455	0.025	0.769
Weddings and Engagements	9	0.639	0.060	0.876	0.462	0.021	0.774	0.462	0.021	0.774	0.502	0.017	0.802

Table 6.3: Schema stability values across four more document-categorical experiments.

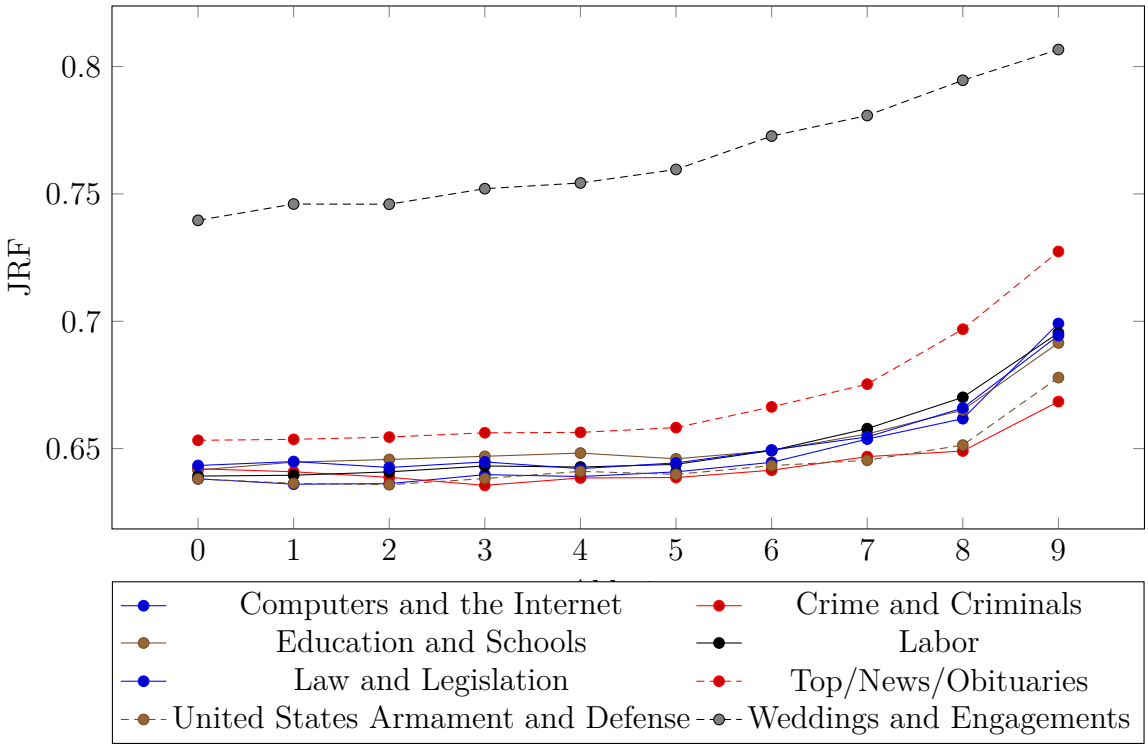


Figure 6.1: Stability averaged across document categories. Ablation is on the x-axis; Jaccard Reciprocal Fraction (e.g. events typically shared) is on the y-axis.

Note that increasing ablation number refers to a decreasing number of documents; in other words, ablation 8 refers to $8/10^{ths}$ of the documents having been *removed*.

In total, the series of experiments generated a total of 3,978,865 schemas, requiring nearly a month to complete. These are not in themselves "unique," as the whole point was to generate schemas that are hopefully as similar to one another as possible. Linear induction produced 2,698,865 schemas, in part a product of its open-ended generation process. Both counter-training and the random walker both individually generated 640,000 schemas, as the number of schemas generated within each category was fixed at 800 for practical computational reasons. [2]

[2]Since the linear induction truncated germinator is simply linear induction but with a few schemas clipped off the end, I do not count this as a separate instance of generation.

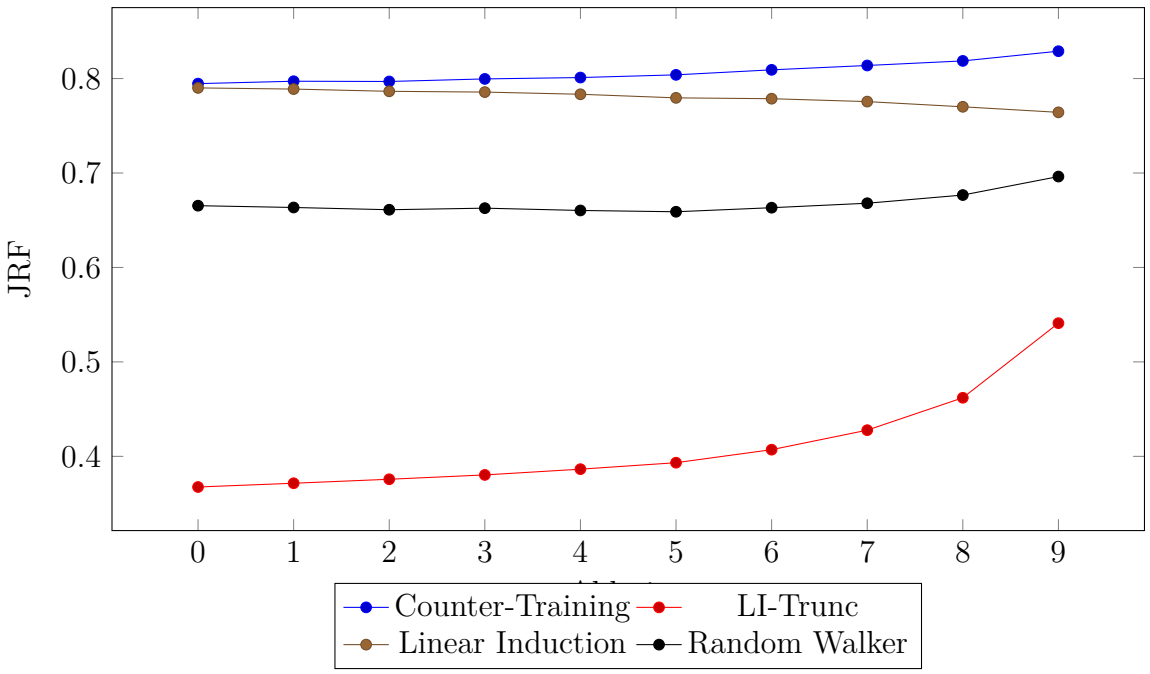


Figure 6.2: Stability performance averaged across germinators. Ablation is on the x-axis; Jaccard Reciprocal Fraction (e.g. events typically shared) is on the y-axis.

Tables (6.2 & 6.3) contain all stability values as computed directly from the resulting schemas. While this has been kept in the dissertation, it is a bit hard to parse on its own. Therefore, I created two figures where I collapsed some dimensions down for better interpretability. Figures (6.1) and (6.2) contain the stability values as averaged across different dimensions. Figure (6.1) averages all stability values for a given ablation across all algorithms, leaving separate values for each category. Conversely, Figure (6.2) averages the stability values across document categories, leaving each algorithm individually expressed.

In Figure (6.1), the document categories found to be homogeneous in Chapter (5) are notably more stable than the categories they found to be heterogeneous. The difference between similarity scores in each cross-validation was significant ($p < 0.001$) in 140/140 comparisons (t-tests) between the similarity scores of "Weddings and Engagements" and the other document categories for both counter-training and the random walker germinators; the difference was significant ($p < 0.001$) in 137/140 comparisons between "Top/News/Obituaries" and the other document categories for both as well. Two of the insignificant differences occurred during the 9^{th} ablation against "Computers and the Internet" and "Labor," one occurred during the 2^{nd} ablation against "Education and Schools." This does not itself have to do with the content of the schemas, but the differences in similarities internal to the document category itself.

In both figures, stability generally increased as the number of total documents decreased. The one exception to this was the linear induction schemas, as shown in Figure (6.2). The causes and consequences of this will be discussed in the next section.

6.4 Discussion

The results in the prior section raise a few questions.

When setting out to run this experiment, I expected stability to increase with the number of documents. The rationale here, as with many statistical learning algorithms, is that the more documents you provide as training data, the better the algorithm does, as it has more examples to leverage and overall improves performance. This presupposes a few things: for example, that better stability is a performance improvement, and that with more data, a germinator can better converge on a single ground truth that reflects the underlying knowledge one has about stories presented in the news.

This did not work out this way in most circumstances. Contrary to expectations, as the number of documents decreases, stability increases, with linear induction constituting the sole exception to this. There are two possible explanations for this: (1) the number of documents withheld in cross-validation decreases with the number of documents, and the stability depends more on the number of documents that differ between cross-validations more than the fraction of documents that differs, or (2) given a fixed number of schemas, they can better capture the information contained in a smaller set of documents because new documents add more novel and unique narratives, thereby making the content of the narratives more difficult to capture in a finite set. In other words, provided with more and more documents, the schemas do not converge to some finite set of schemas but instead are presented with an increasingly difficult problem to solve. These two explanations are not completely orthogonal; however, the first is a far more mechanical explanation. It is simply that more word types are contained in a larger set of documents, and therefore will be subject to a larger Zipfian tail, which is more difficult for a finite set of recorded event verbs to

cover. The second presupposes that the system has some semblance of understanding the language data and narrative, and the challenge comes from the increased diversity of narratives contained therein. The second explanation could cause the first, but the second claim requires a much greater burden of evidence because of its deeper implications.

Contrary to the other germinators, linear induction has no limit to the number of schemas it can produce, albeit with a great number of schemas containing single events. This means there are two possible explanations for its behavior. The first, the explanation originally hoped for—that as the algorithm is given more documents, it begins to converge on a stable core of knowledge derived from the source data that in one form or another represents a consistent understanding of the data. The second explanation is more mundane—that what's actually stable is the schemas containing single events, schemas which then bias the apparent stability upward as more events that do not cross the β threshold are observed in a greater number of documents.

The LI-Truncated stability provides some insight here. If there is a stable content core that linear induction is approaching, then the first 800 schemas generated should reflect that. What we see instead is the lowest stability scores across the board. However, note that as ablation 9 is approached, the stability of the LI-Truncated schemas increases, likely due to the fact that the number of schemas actually produced by linear induction is approaching 800, so the effect of the truncation is beginning to wear off.

While the arbitrary clipping of the linear induction schemas greatly decreases stability, this may point to the unexpected decrease in stability in the other germinators. Given their hard limit on the number of schemas used, they are in some sense performing a similar "clipping" of the content contained in the documents. However,

their increased stability, while still conducting a form of clipping, could be attributed to performing a more informed clipping than the LI-Truncated schemas.

Not all here is surprising. Meeting expectations, the Weddings and the Obituaries categories were consistently more stable at all ablations than their heterogeneous counterparts, as expected given the findings in Chapter (5). The differences between were significantly different the vast majority of the time (277/280), and in cases where this was not the case, only a handful of the stability scores of the heterogeneous categories had increased. Additionally, the gap between Weddings and Engagements and the Obituaries sections stability is comparable to the performance difference observed previously in the NASTEA task.

6.5 Conclusions

In this chapter, I explored the stability of narrative schemas. I used both an ablation and cross-validation of the data to produce different sets of schemas and compare them using the Fuzzy Jaccard coefficient and the Jaccard Reciprocal Fraction, which produces a transformation of the Fuzzy Jaccard coefficient that is easier to interpret.

Overall, with respect to document categories, the homogeneous categories produced more stable batches of schemas than the heterogeneous ones. The counter-training and linear induction germinators produced more stable results, but the linear induction truncated results indicate that much of the apparent stability of the linear induction germinator is contained in its long tail of schemas.

Additionally, contradicting expectations, the schemas produced were more stable when given fewer documents. It is difficult to say whether this is because fewer documents were removed at every step of the cross-validation or because there is simply fewer narratives to capture in the same number of schemas. These explanations do

not necessarily contradict one another, and the effect witnessed may be a mixture of both.

Understanding the stability of schemas provides us with a window into the quantitative properties of news narratives at large. Additionally, stability provides another technique for evaluating what makes a "good set of schemas:" as objects that are consistent regardless of what subset of a corpus they are derived from. Not for all purposes is this property necessarily "good," but it does provide a quantitative metric for this notion.

In the next chapter, I conclude this dissertation.

Chapter 7

Conclusions

In this chapter, I conclude this dissertation.

In Section (7.1), I summarize the findings of the dissertation. In Section (7.2), I reflect on those findings, my goals at the outset of undertaking the dissertation, and how the discrepancies between those two suggest new methodologies for conducting novel research using unsupervised learning. In Section (7.3), I discuss future work derived from that in this dissertation.

7.1 Summary of Findings

In Chapter (3), I presented the distinction between *score*—the relationship between a schema and a candidate event—and *germinators*, the way in which a score is interpreted and candidates are traversed. I re-interpreted Chambers & Jurafsky (2009)'s technique for generating schemas in these terms, and in doing so devised the Linear Induction (LI) germinator. In addition, I devised two other new germinators, the counter-training (CT) technique, inspired by Yangarber (2003), and the random walker (RW) technique. All of these have unique properties that produce different, distinct sorts of schemas.

Schemas themselves are not directly evaluated by the frequently used *cloze task*; in fact, the dividing line between score and germinator is where the cloze task is executed. Therefore, in Chapter (4), I also present two new evaluation techniques.

Both of these rely on a *presence measure* to work, which determines between a schema and a document how "present" the schema is in the document based on the density and dispersion of its events through the document. Presence is used by both evaluations demonstrated here: the *NASTEA* task and the *MDL* measures.

In the *NASTEA (Narrative Argument Salience Through Entities Annotated) task*, a set of schemas are used to extract salient entities—as annotated in the NYT corpus (Sandhaus, 2008a)—from news text. The intuition here is that good schemas should extract the same entities an outside annotator deemed relevant from the text.

On the other hand, the *MDL (minimum description length) measures* give an information theoretic approach to examine a set of schemas. One of these measures is a *model size* measure, denoted $|M|$. This measures in formal terms the size of a model M in nits. The other measure is *corpus size* or $|C|_M$, which measures how well a model M can encode the events contained in a corpus C. Both of these should be as small as possible.

For a first pass, in Chapter (4), all three germinators score in the high 30% F1 on the NASTEA task, while their $|C|_M$ values are around $36,000$, except for the RW schemas which scored around 35,000 and were significantly different from the others—however, they managed to do this while obtaining a model size that was around 25% larger than the others.

I further optimized for performance on the NASTEA task, performing a parameter climb to optimize performance. In the dev portion, the RW schemas attained peak performance with an F1 of 0.422, but dropping a staggering 1.5% on test. CT schemas performed the worst during dev but had the best test performance, only dropping 0.1% between dev and test. The robustness of the CT schemas is surprising, but bodes well for their potential use in applications.

To evaluate the MDL measure, I used the OntoNotes corpus to generate schemas with both OntoNotes' gold standard annotations and a set of automatic annotations. Although the dataset is relatively small, I expected the automatic annotations to perform worse on the MDL measure because automatic coreference is notoriously poor. However, instead, the MDL measure insignificantly degraded when the gold standard annotations were used. This was shown to likely be caused by the automatic annotations successfully pulling in more events to add to schemas—technically correct or not—than the automatic annotations.

Chapter (5) analyzed the relationship between narrative schemas and document category. By document category, I am referring to those annotated in the `onlineproducer` data of the NYT Corpus Sandhaus (2008a). First, I use narrative schemas to classify documents. This did not work well, with the best classification achieving an F1 of 0.461. This seemed to entail that schemas do not predict document categories well, at least without potentially considering their argument types.

Second, I used document categories to generate different sets of narrative schemas and analyze their performance on the NASTEA task. This resulted in some discovering that, with respect to the distribution of narrative schemas within them, there are roughly two types of document categories: homogeneous and heterogeneous. *Homogeneous* document categories have concave up or flat NASTEA curves and perform best at N_1—that is, using only the single schema that scored the highest presence in a document to extract entities. They are categories that are written from templates—*Weddings* and *Obituaries*—so schema extraction thrives on them. Most typical news categories are *heterogeneous*, however. Their NASTEA curves are concave down and perform best at N_{11} or later, and they require often many schemas applied to a document to achieve optimal F1 performance in the NASTEA task. This seems to indicate

that they are constructed from many fragments of different stories, parts of which may be systematic or entail pieces of recurring events, but on the whole are not repetitive.

To finish Chapter (5), I test whether topics can be introduced as substitutes for document categories. Because document categories are unavailable in new applications but topic models can be learned in an unsupervised manner, there was a great potential benefit to adding topics to the schema generation model. I devised a technique for assigning topic weights to a schema via a *pseudodoc*. However, the schemas generated did not seem to yield a performance boost on the NASTEA task.

In Chapter (6), I perform an ablation and cross-validation of the same eight document categories examined in Chapter (5) to determine how stable narrative schemas are under perturbations to the document set and as documents are removed. The expectation here was that an increase in the number of documents would increase the stability of the schemas. In fact, it turned out schemas were more stable when fewer documents were provided as training data. The likely explanation for this is that more documents simply mean more narratives to contain in the same sized space. Unlike other applications in statistical NLP, more data does not simply provide more examples of things likely to happen again, but instead gives examples of more novel situations, leading to divergence instead of convergence.

I will take a broader look at these results and their consequences through the rest of this chapter.

7.1.1 Core Take-Aways

The goal here originally was to conduct an unsupervised analysis of text using narrative schemas. I did not reach that goal, but I hope that some of the properties and phenomena determined here can help anyone (including myself!) in the future who seeks to do such an analysis.

Many of the things I tried in this dissertation were intended to be the simplest possible approach to the problem they sought to solve because no one had quite looked at the problem before. The next step is to add the appropriate complications. Those complications depend on the findings of the simpler approaches.

Homogeneous / Heterogeneous Categories

The quantitative determination that two types of narratologically distinct document category types exist is a first. While such a distinction may have been made qualitatively previously, the techniques employed in Section (5.3) show quantitatively that such a distinction is measurable; this is further reinforced by the results in Chapter (6).

Most news document categories are heterogeneous. Any technique that considers the presence and appearance of discrete schemas in a document should factor in that most documents are composed of *multiple* schemas patched over one another, not just one.

Schema-Schema and Schema-Document Measures

Both schema presence and the Fuzzy Jaccard / Jaccard Reciprocal Fraction proved useful tools throughout the dissertation. Though presence is not perfect—having been built on a lot of assumptions about documents being narratologically homogeneous—it is a good first step and is readily applicable to other problems that leverage schemas. With some modifications, it could be better suited to heterogeneity as well, not just scoring a single schema to the document as a whole, but yielding a set of schemas that sufficiently cover a document.

Topics Are Not Document Categories

From the way topics (Blei et al., 2003) are often discussed, they are treated as an unsupervised substitute for document categories. In the model used in Section (5.4),I found that LDA did not really seem to help much directly with schema generation, though this model did not work exactly the same way as document categories. Such a model, where topics are siloed from one another is plausible, though, although the document category-topic fit is not exactly one-to-one: for example, the two homogeneous document categories were combined and some document categories were sort of blurry mixtures of many different topics. Most heterogeneous categories had one document category into which most of their documents fit, though. Perhaps with more finagling it might be possible to use them for such a purpose, but this does not seem to be an easy out-of-the-box problem to address.

Be Wary of the One-Trick Event Pony

In many ways, the evaluations considered in Chapter (4) felt unsatisfying. The core reason for this sensation, I suspect, was that despite vast differences in sets of schemas, the scores reflected little difference amongst them. This suggests that the scores do not reflect the differences in schemas as hoped would happen.

Using events as the central and whole focus of studies of narrative is not really a good idea. Events are only one piece of the puzzle. The OntoNotes experiments showed this—by overly focusing on events, worse data was able to score about the same. NASTEA pushes things in the right direction, but as it is implemented here, is far from perfect. If considering events, make sure to look at other elements too. The classification-with-schemas experiments showed that, at the least, the presence of specific argument slots can indicate different document categories with high precision,

and specific types might improve that even further. While this was done here, moving forward, I cannot really see any way around employing measures that take more into account other than the events of a schema.

Narratives Will Never Converge

Thus far, the evidence points to narrative schemas being Zipfian—or at least, the types of narratives from which they are derived increase without bound. It is possible that this is the driver behind the long tail of the power law distribution of tokens—as we have more stories to tell, we need more words to tell them. In either case, the stability experiments suggest that the narratives contained in text seem to increase without bound. There will be new narratives until there is no one left to live them; it is impossible to live the exact same story twice, but many narratives will be told with the fragments of others, and some of those fragments just happen to be nearly the same size and content as fragments that already happened.

7.2 Reflections on Original Goals

At the outset of this dissertation, the goal was to conduct an unsupervised analysis of a previously unseen corpus of documents using complex, discourse-level structures. This, however, became bogged down in one of the fundamental goals: evaluation. Evaluation is considered essential in any NLP task, and for good reason. If one purports to be doing something, without evaluation, it is not clear that thing is actually being done.

Most of this dissertation is a series of attempts to evaluate in a strict engineering sense the performance or quality of narrative schemas. But more important than the performance in any formal sense—when attempting to conduct an unsupervised

analysis—are the properties of the schemas themselves and whether those properties are desirable in a given analytical scenario. This is true of models more generally when used for unsupervised analysis of linguistic data. It is not the strict performance on an evaluation metric that counts, but the properties of the model that tell us whether that model is good or bad.

Evaluation metrics can get at such properties, but they do not always. Especially for linguistic objects as complex as schemas, a simple numerical metric over-simplifies in many circumstances.

An alternative point of view presents schemas not as a object of performance, but an object of analysis in its own right, distilling the content of often complex linguistic data set into pieces that are systematic and meaningful. By systematic, I mean created in a way that is quantitative, reproducable, and follows a series of justifiable choices in its construction. By meaningful, I mean that it is brings together components of the real world to an end user that confirm their own qualitative observations or provide them with new observations that would have otherwise been unobtainable with their available time or resources.

In neither of these cases to we care that schemas are "good," only in the sense that they do reflect the real world in one way or another. In other words, the question is not how good or bad the schemas are, but what do the schemas tell us about the data. In this dissertation, I continuously struggle with the former in pursuit of the latter, but it's not quite clear that the former is relevant or the right question to be asking in an unsupervised study. There's no way to know whether the product of an unsupverised study on a new data set is "good;" rather, we need to know reliably the properties of the system applied to the data so that we understand the relationship between the model, its products, and the real world.

In that sense, in all circumstances in developing artificial intelligence applications, performance on a dataset is merely one *property* of the algorithm being applied. This works for a lot of the problems that have been the focus of NLP for the last 30+ years, but it selects for problems that lend themselves to single metric analysis at the exclusion of problems for which understanding their general properties is more insightful.

Schemas are a domain where the output itself requires its own analysis. This makes it difficult for them to fit into a typical generate-evaluate 8-page ACL paper narrative. At the same time, such an analysis is often too technical for most venues interested in the sorts of questions they could be applied to. Finding a way to present such results in non-technical venues or changing the traditional NLP venues expectations beyond mere engineering to allow for deep analysis will be necessary to allow work to be published moving forward.

What to do with this new found knowledge? I address this in the next section.

7.3 Future Work

While this dissertation asked and gave rudimentary answers to a number of elementary questions about narrative schemas, there are many nuanced corners and questions still to ask. While there is a lot of content here, in many respects the experiments in this dissertation raised more questions than answers. Many of these future projects are ideas that were originally slated for this dissertation before time ran out, and it became clear they were whole projects in themselves.

I've divided these into two subsections, improvements to schemas and the measures used to understand their properties (Section 7.3.1) and potential uses for schemas in other applications (Section 7.3.2).

7.3.1 Improvements to Schemas and their Evaluations

In this section, I discuss more technical aspects of how schemas might be improved, either in capturing more information about the language data or in their evaluation.

Heterogeneous MDL and Schema Presence

The presence measure and the MDL measure for $|C|_M$ defined in Chapter (3) are inherently homogeneous. $|C|_M$ tries to apply a single schema to "encode" the events in each document and then penalizes for further errors; the findings in Chapter (5) show quite clearly that multiple schemas will be needed to cover whole documents in most document categories. Extending them for use in this heterogeneous reality, where multiple schemas may encode a document, would change the picture they paint of schemas tremendously.

Ordering of Events

Chambers & Jurafsky (2008) experimented with ordering events in chains. However, Chambers & Jurafsky (2009) largely abandoned the partial ordering attempted in prior work. Jans et al. (2012) showed that text ordering can improve performance on the cloze task. Different sorts of ordering can occur as well—events can re-occur and events can partially or totally overlap one another, to the point that some events can have sub-events. Adding ordering to schemas could yield potentially interesting results, though I suspect it would largely add sparsity to the existing data. As a result, this would improve precision and lower recall on the NASTEA task, as the included ordering would make the entity extraction pickier about selecting documents. Whether this effect would result in an overall improvement in F1 scores is largely an

empirical question, and special, new evaluations might be required to evaluate this sort of ordering.

Toward More Sophisticated Schemas

Any of a number of linguistic phenomena—not even all enumerated here—could be added to narrative schemas to potentially improve their performance on various evaluation metrics and/or to extend their capabilities. I have broken these down into three possibilities to look at: lexical presence, lexical interpretation, and extended structure.

The most rudimentary extraction available is the presence or absence of a particular sort of lexical presence feature that has already been identfied by existing NLP tools—for example, to extract from a dependency parse the presence of modality and negation in a sentence—and must only be integrated into a new schema model. These could be added relatively easily to existing schema models. However, their interpretation is unclear. For example, if I have to tell you that something explicitly did *not* happen, is that because it is something that never happens, or because it is something that implicitly should have happened, based on your schemas about the world, and I need to cancel that out? The same goes for modality—the pragmatics of modality and negation lend themselves to being overt cancellations of implications we make based on script-knowledge and suggest that they could be strong features for inferring schemas.

Both negation and modality act as new lexical presence features to extract and add to schemas, but what about issues of polysemy and synonymy, problems with the words we've already extracted? These are largely problems of lexical interpretation—in other words, once information about a feature is extracted, how can that feature be best interpreted? How can lexical information—say, provided by word2vec (Mikolov

et al., 2013)—be better leveraged for solving these issues? Adding a type of scoring or germinator that takes such measures of semantic similarity could greatly improve schemas by more carefully selecting events to add to schemas. Additionally, identification of light verbs could help improve the set of events under consideration, treating both the verb and its object—instead of the verb alone—as an event.

Furthermore, new sorts of relationships—other than simply sharing entity slots—could be retained in schemas. For example, sets of schemas often contain duplicate schemas. One potential solution is to merge schemas that contain nearly similar sets of events. This merging process brings a lot of potential complexities into the process—for example, do chains of differing event represent a fork in the series of events, or simply an event that was previously missed? How could this be decided? What about events that are directly contradictory? For example, if someone is wounded that implicates that they were not killed, but schemas that involve "wound" often involve "kill," both sharing the same `OBJ` slot despite the implications of one being mentioned over the other. Schemas that contain the logical structure of events could start to have a broad use of applications in correctly determining entailments based on story logic. This is arguably a much harder piece of information to tease out than simply extracting indicators of modality from a sentence. How to get at these threads of possibility is an open-ended question. Similarly, shared referent is not the only possible relation to structure a schema around. Schemas could also be generated with discourse relations other than that they shared entity slots, such as those encoded in the Penn Discourse Treebank (Prasad et al., 2007), present an open possiblity for research. In conjunction with more complex models of schema structure, such relations could appear side-by-side with a shared entity slot model. Answering many of these questions would each require new experiments and evaluations with respect to schema structure. Additionally, integrating these different layers together may itself require

careful consideration to properly encapsulate their interactions. However, the answers to these questions may prove interesting—understanding the logic behind these sorts of interactions is often abstracted or implied in linguistic research. Finding clear answers to these questions could provide insightful in understanding the foundations of language.

Broader Extraction

While an *event* in this dissertation was used almost interchangeably with *verb*, the two are not necessarily interchangeable. For example, consider the following quote:

> "The Shah's army was *split* by diverse internecine feuds and by the Shah's decision to divide his army into small groups concentrated in various cities. *This fragmentation* was decisive in Khwarazmia's defeats..." — The Wikipedia Article on Genghis Khan[1]

Here, "this fragmentation" refers to the "split" mentioned in the previous sentence. However, nothing in the second mention of the same event would contribute to information linked to that event in the schema; the current paradigm ignores nominal events. Generalizing extracted events could lead to better, broader, and richer schemas that obtain more information from their source material.

A Better NASTEA

The NYT Corpus' salient entities used in the NASTEA task were largely selected out of convenience. Better would be a corpus with annotations more oriented for narrative, and with more principles selections of entities. Some such corpora have

[1] `https://en.wikipedia.org/wiki/Genghis_Khan`, last edited 22 October 2017, at 12:53

been planned or annotated since this dissertation was undertaken (Caselli & Vossen, 2016; Modi et al., 2017), and leveraging them as an evaluation tool could provide a much more accurate salient entity extraction, much more insight for improvement, and potentially open other avenues to similar evaluations beyond NASTEA that examine other properties of schemas and their relationships to the aspects of texts they should reflect.

7.3.2 Applications of Schemas

The previous section suggested many ways in which schemas and assessments of their properties can be potentially improved. However, schemas are generated with a potential goal in mind—some practical circumstance where they may be employed.

In this section, I discuss three possible applications of schemas: analysis of text, using them to improve NLP tools, and potentially for improving machine translation.

Analysis of Text

Of course, this dissertation set out to generate schemas for this purpose, the goal here being to reflect underlying structures in text that are themselves difficult to tease out of text *en masse*.

But what good would those structures be? For example, as a tool for alignment, schemas might provide a good way to compare where similar narratives are discussed in different texts.

As an object of analysis, the comparison of schemas discussing similar individuals or topics may provide useful for understanding generally how such narratives could be discussed across news sources or periods of time.

Just as important, devising a technique for consolidating the properties of narrative schemas into a measure of certainty about a specific result—statistical or

otherwise—could prove extremely useful. Indeed, such a technique for determining the certainty of the outputs of any unsupervised learning technique could generalize beyond schemas to enable more broad applications of unsupervised learning with a better understanding of their quality.

To Improve NLP Tools

Generally, schemas may contribute to improvements on these tasks by establishing expectations of the sort that humans have, allowing for cases that are genuinely ambiguous to be resolved with the world knowledge that schemas can provide in a structured way.

For example, in a schema-informed parser, PP attachment may be improved. A particular event verb might expect a `PREP` argument of a certain type—if the object of a specific preposition is that type, then the schema can help resolve that ambiguity. Similarly, coreference may improve with a set of schemas helping to smooth the extraction process. Since the schemas have expectations about which slots across events have coreference links, the schemas can help resolve those expectations in difficult cases.

Of course, there is a sort of chicken-and-egg problem with these kinds of applications. Schemas as they are derived now are based on automatically parsed and coreferenced text. Adding schemas that might be erroneous because of some systematic error may reinforce that error in the original systems. That means that for such an improvement to be implemented, the schemas or the original systems must both be of a high enough quality already that the improvements outweigh any damage done.

Machine Translation

Rather than purely token-driven alignments, schema-driven alignments across text could greatly improve translation by allowing for alignments of gestalts across languages—gestalts in the sense of collections of events which may be translated together as a group in a way that single token alignments struggle to provide.

7.4 Wrap Up

The questions this dissertation started with are largely still open. In trying to answer those questions, many more gaps in knowledge about schemas became apparent and needed to be answered in order to pursue the original questions.

In fact, even with the experiments undertaken here, there is still much to learn about schemas and the broader understanding of text, with many directions to explore. Covering all possibilities in this text would have been impossible.

I hope that some of these properties and the approaches taken to assessing them will prove useful to those attempting to understand text at a discourse level—quantitatively or qualitatively—in the future, even if the approaches taken do not rely on discrete content schemas. While working with more complex outputs from text—like schemas—is difficult, the promise of better understanding the foundations of knowledge is all the worthwhile and inescapable. These problems will have to be solved in one way or another to develop more sophisticated artificial intelligence that can better interact with and properly interpret the world it exists in. The discoveries made here help narrow the scope of those problems; however, the scope of those problems is incomprehensibly massive. Relatively speaking, this work only begins here.

Bibliography

AlMalek, J. S. (2017). *The Youngest ISIS child Suicide bomber captured in Iraq.* Retrieved from `https://www.liveleak.com/view?i=036_1490167456`

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley Framenet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1* (pp. 86–90).

Bal, M. (1997). *Narratology: Introduction to the theory of narrative.* University of Toronto Press.

Balasubramanian, N., Solderland, S., Mausam, & Etzioni, O. (2013). Generating coherent event schemas at scale. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1721–1731). Association for Computational Linguistics.

Bentivogli, L., & Giampiccolo, D. (2011). *Seventh Recognizing Textual Entailment Challenge at TAC 2011.* National Institute of Standards and Technology. Retrieved from `https://tac.nist.gov//2011/RTE/RTE7_CFP.txt` (Accessed: 2017-03-23)

Bird, S., Loper, E., & Klein, E. (2009). *Natural Language Processing with Python.* O'Reilly Media Inc.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022.

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

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, *30*(1), 107–117.

Caselli, T., & Vossen, P. (2016). The Storyline Annotation and Representation Scheme (StaR): A Proposal. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)* (pp. 67–72). Austin, Texas: Association for Computational Linguistics. Retrieved from `http://aclweb.org/anthology/W16-5708`

Chambers, N. (2011). *Inducing event schemas and their participants from unlabeled text.* (Unpublished doctoral dissertation). Stanford University.

Chambers, N. (2013). Event schema induction with a probabilistic entity-driven model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1797–1807). Association for Computational Linguistics.

Chambers, N., & Jurafsky, D. (2008). Unsupervised learning of narrative event chains. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)* (pp. 789–797). Association for Computational Linguistics.

Chambers, N., & Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation for Natural Language Processing.* (pp. 602–610). Association for Computational Linguistics.

Chambers, N., & Jurafsky, D. (2011). Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (pp. 976–986).

Cheung, J., Poon, H., & Vanderwende, L. (2013). Probabilistic frame induction. In *Proceedings of the Conference of the North American Association for Computational Linguistics and Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics.

Chinchor, N. A. (2001). *Overview of MUC-7/MET-2.* Retrieved 2016-10-14, from `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_proceedings/overview.html`

Creutz, M., & Lagus, K. (2002). Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning* (pp. 21–30). Association for Computational Linguistics. Retrieved from `http://www.aclweb.org/anthology/W02-0603` doi: 10.3115/1118647.1118650

Creutz, M., & Lagus, K. (2004). Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology* (pp. 43–51). Barcelona, Spain: Association for Computational Linguistics.

DeJong, G. (1983). Acquiring schemata through understanding and generalizing plans. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (Vol. 51).

de Marneffe, M., MacCartney, B., & Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the Language*

Resources and Evaluation Conference of the European Language Resources Association (LREC). Association for Computational Linguistics.

Elson, D. K., Dames, N., & McKeown, K. R. (2010). Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 138–147). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from `http://dl.acm.org/citation.cfm?id=1858681.1858696`

Elson, D. K., & McKeown, K. (2009). Extending and evaluating a platform for story understanding. In *AAAI Spring Symposium: Intelligent Narrative Technologies II* (pp. 32–35).

Elson, D. K., & McKeown, K. R. (2007). A platform for symbolically encoding human narratives. In *Proceedings of the AAAI Fall Symposium on Intelligent Narrative Technologies.*

Fellbaum, C. (1998). *WordNet.* Wiley Online Library.

Fillmore, C. J., Johnson, C. R., & Petruck, M. R. (2003). Background to Framenet. *International Journal of Lexicography*, *16*(3), 235–250.

Finkel, J., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (pp. 363–370).

Goddard, C., & Wierzbicka, A. (1994). *Semantic and lexical universals: Theory and empirical findings* (Vol. 25). John Benjamins Publishing.

Goldsmith, J. (2006). An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, *12*(04), 353–371.

Green, R., & Dorr, B. (2004). Inducing a semantic frame lexicon from WordNet data. In *Proceedings of the 2nd Workshop on Text Meaning and Interpretation* (pp. 65–72). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=1628275.1628284

Grishman, R., & Sundheim, B. (1996). Message Understanding Conference-6: A Brief History. In *COLING* (Vol. 1, pp. 466–471).

Hall, D., Jurafsky, D., & Manning, C. D. (2008). Studying the history of ideas using topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 363–371).

Haveliwala, T. H. (2002). Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web* (pp. 517–526).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computing*, *9*(8), 1735–1780. Retrieved from http://dx.doi.org/10.1162/neco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management* (pp. 2333–2338).

Jans, B., Bethard, S., Vulić, I., & Moens, M. (2012). Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 336–344).

Kilgarriff, A. (1997). I don't believe in word senses. *Computers and the Humanities*, *31*(2), 91–113.

Lakoff, G., & Johnson, M. (1980). *Metaphors we live by.* University of Chicago press.

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., & Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, *39*(4), 885–916.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*(Apr), 361–397.

Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing.* MIT Press.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55–60).

Marsh, E., & Perzanowski, D. (1998). *MUC-7 Evaluation of IE Technology: Overview of Results.* National Institute of Standards and Technology.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *27th Annual Conference on Neural Information Processing Systems (NIPS)* (pp. 3111–3119). Curran Associates.

Miller, B., Olive, J., Gopavaram, S., & Shrestha, A. (2015). Cross-document non-fiction narrative alignment. In *Proceedings of the First Workshop on Computing*

News Storylines (pp. 56–61). Beijing, China: Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/W15-4509

Miller, G. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, *63*(2), 81.

Minsky, M. (1974). A framework for representing knowledge. Retrieved from http://hdl.handle.net/1721.1/6089

Mitchell, T., Betteridge, J., Carlson, A., Hruschka, E., & Wang, R. (2009). Populating the semantic web by macro-reading internet text. *The Semantic Web-ISWC 2009*, 998–1002.

Mnih, A., & Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning* (pp. 641–648).

Modi, A., Anikina, T., Ostermann, S., & Pinkal, M. (2017). Inscript: Narrative texts annotated with script information. *arXiv preprint arXiv:1703.05260*.

Mooney, R., & DeJong, G. (1985). Learning schemata for natural language processing. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 681–687).

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., ... Allen, J. (2016). A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Mostafazadeh, N., Roth, M., Louis, A., Chambers, N., & Allen, J. (2017). *LSDSem 2017 Shared Task: The Story Cloze Test"*. Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/W17-0906

Nguyen, K.-H., Tannier, X., Ferret, O., & Besançon, R. (2015). Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics (ACL-15)* (pp. 188–197).

NIST. (2014). *Text REtrieval Conference (TREC) Frequently Asked Questions.* http://trec.nist.gov/faq.html. (Accessed: 2016-10-17)

Norvig, P. (1983). Frame activated inferences in a story understanding program. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 624–626). Retrieved from `http://norvig.com/ijcai83.html`

Ogawa, H., Nishi, J., & Tanaka, K. (1980). The knowledge representation for a story understanding and simulation system. In *Proceedings of The 8th International Conference on Computational Linguistics (COLING 80).*

Owczarzak, K., & Dang, H. T. (2010). TAC 2010 Guided Summarization Task Guidelines. Retrieved from `http://tac.nist.gov/2010/Summarization/Guided-Summ.2010.guidelines.html`

Özbal, G., Strapparava, C., Tekiroglu, S. S., & Pighin, D. (2016). Learning to identify metaphors from a corpus of proverbs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 2060–2065). Austin, Texas: Association for Computational Linguistics. Retrieved from `https://aclweb.org/anthology/D16-1220`

Pantel, P., & Ravichandran, D. (2004). Automatically labeling semantic classes. In S. Dumais, D. Marcu, & S. Roukos (Eds.), *HLT-NAACL 2004: Main Proceedings* (pp. 321–328). Boston, Massachusetts, USA: Association for Computational Linguistics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Pichotta, K., & Mooney, R. J. (2014). Statistical script learning with multi-argument events. In *EACL* (Vol. 14, pp. 220–229).

Pichotta, K., & Mooney, R. J. (2015). Learning statistical scripts with lstm recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence.*

Pichotta, K., & Mooney, R. J. (2016). Using sentence-level lstm language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 279–289). Berlin, Germany: Association for Computational Linguistics. Retrieved from `http://www.aclweb.org/anthology/P16-1027`

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R., & Xue, N. (2011). CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task* (pp. 1–27). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from `http://dl.acm.org/citation.cfm?id=2132936.2132937`

Prasad, R., Miltsakaki, E., Dinesh, N., Lee, A., Joshi, A., Robaldo, L., & Webber, B. L. (2007). The Penn Discourse Treebank 2.0 Annotation Manual.

Řehůřek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges*

for NLP Frameworks (pp. 45–50). Valletta, Malta: ELRA. (http://is.muni.cz/publication/884893/en)

Reiter, N., Frank, A., & Hellwig, O. (2014). An NLP-based cross-document approach to narrative structure discovery. *Literary and Linguistic Computing*, *29*(4), 583–605.

Rich, E., & Knight, K. (1991). *Artifical intelligence*. New York: McGraw-Hill.

Rossum, G. (1995). Python reference manual. Retrieved from http://www.python.org

Rudinger, R., Demberg, V., Modi, A., Van Durme, B., & Pinkal, M. (2015). Learning to predict script events from domain-specific text. In *Lexical and Computational Semantics (* SEM 2015)* (p. 205).

Rudinger, R., Rastogi, P., Ferraro, F., & Van Durme, B. (2015). Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-15).*

Ruppenhofer, J., Ellsworth, M., Petruck, M. R., Johnson, C. R., & Scheffczyk, J. (2006). *FrameNet II: Extended theory and practice*. (Revised November 1, 2016)

Sammons, M., Vydiswaran, V. V., & Roth, D. (2012). *Recognizing textual entailment*. Urbana, IL: University of Illinois. Retrieved from http://l2r.cs.uiuc.edu/~danr/Teaching/CS546-12/TeChapter.pdf

Sandhaus, E. (2008a). The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, *6*(12), e26752.

Sandhaus, E. (2008b). The New York Times Annotated Corpus Overview. *New York Times Research and Development*. Retrieved from https://catalog.ldc.upenn.edu/docs/LDC2008T19/new_york_times_annotated_corpus.pdf

Schank, R. (1973). Identification of conceptualizations underlying natural language. In R. Schank & K. Colby (Eds.), *Computer Models of Thought and Language.* San Francisco: Freeman.

Schank, R., & Abelson, R. (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures.* New Jersey: Lawrence Erlbaum.

Schwartz, R., Sap, M., Konstas, I., Zilles, L., Choi, Y., & Smith, N. A. (2017). *Story Cloze Task: UW NLP System.* Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/W17-0907

Sim, Y., Routledge, B., & Smith, N. A. (2016). Friends with motives: Using text to infer influence on SCOTUS. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 1724–1733). Austin, Texas: Association for Computational Linguistics. Retrieved from https://aclweb.org/anthology/D16-1178

Simonson, D., & Davis, A. (2016). NASTEA: Investigating narrative schemas through annotated entities. In *Proceedings of the Second Workshop on Computing News Storylines (CNS 2016) at EMNLP 2016* (pp. 57–66). Austin, Texas: Association for Computational Linguistics. Retrieved from http://aclweb.org/anthology/W16-5707

Strummer, J. (1982). *Know your rights.* London, New York City, Warnford. (Track 1 of Combat Rock by The Clash)

Taylor, W. L. (1953). "Cloze procedure": a new tool for measuring readability. *Journalism Bulletin*, *30*(4), 415–433.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1* (pp. 173–180).

Toutanova, K., & Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13* (pp. 63–70).

Tran, T., & Ostendorf, M. (2016). Characterizing the language of online communities and its relation to community reception. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 1030–1035). Austin, Texas: Association for Computational Linguistics. Retrieved from `https://aclweb.org/anthology/D16-1108`

United States Defense Advanced Research Projects Agency. Software and Intelligent Systems Technology Office. (1992). *Fourth Message Understanding Conference, (MUC-4): Proceedings of a Conference Held in McLean, Virginia*. Morgan Kaufmann Publishers. Retrieved from `https://books.google.com/books?id=X6jgAAAAMAAJ`

Vossen, P., Caselli, T., & Kontzopoulou, Y. (2015). Storylines for structuring massive streams of news. In *Proceedings of the First Computing News Storylines*

Workshop (CnewS) at ACL-IJCNLP 2015 (pp. 40–49). Beijing, China: Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/W15-4507

Wallach, H. M. (2011). *Statistical topic models for computational social science.* Retrieved from https://people.cs.umass.edu/~wallach/talks/2011-04-05_CLSP.pdf (Slides of a talk given at Johns Hopkins Univerity Center for Language and Speech Processing)

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., ... Houston, A. (2013). OntoNotes Release 5.0. *Linguistic Data Consortium, Philadelphia*(LDC2013T19).

Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning.* Advanced Book Program, Addison-Wesley Pub. Co., Reading, MA.

Wilson, S., Mihalcea, R., Boyd, R., & Pennebaker, J. (2016). Disentangling topic models: A cross-cultural analysis of personal values through words. In *Proceedings of the First Workshop on NLP and Computational Social Science* (pp. 143–152). Austin, Texas: Association for Computational Linguistics. Retrieved from http://aclweb.org/anthology/W16-5619

Wodak, R., & Weiss, G. (2003). *Critical Discourse Analysis: Theory and Interdisciplinarity.* Palgrave Macmillan.

Wu, Z., & Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics* (pp. 133–138). Stroudsburg, PA, USA: Association for Computational

Linguistics. Retrieved from https://doi.org/10.3115/981732.981751 doi: 10.3115/981732.981751

Yangarber, R. (2003). Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1* (pp. 343–350). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dx.doi.org/10.3115/1075096.1075140 doi: 10.3115/1075096.1075140

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics* (pp. 189–196).

Zaenen, A., Karttunen, L., & Crouch, R. (2005). Local textual inference: Can it be defined or circumscribed? In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment* (pp. 31–36). Ann Arbor, Michigan: Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/W/W05/W05-1206

Zaremba, W., & Sutskever, I. (2014). Learning to execute. *CoRR, abs/1410.4615*. Retrieved from http://arxiv.org/abs/1410.4615