INDUCING EVENT SCHEMAS AND THEIR PARTICIPANTS
FROM UNLabeled TEXT

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Nathanael William Chambers
May 2011
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daniel Jurafsky, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christopher Manning

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Andrew Ng

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.
Abstract

The majority of information on the Internet is expressed in written text. Understanding and extracting this information is crucial to building intelligent systems that can organize this knowledge, but most algorithms focus on learning atomic facts and relations. For instance, we can reliably extract facts like “Stanford is a University” and “Professors teach Science” by observing redundant word patterns across a corpus. However, these facts do not capture richer knowledge like the way detonating a bomb is related to destroying a building, or that the perpetrator who was convicted must have been arrested. A structured model of these events and entities is needed to understand language across many genres, including news, blogs, and even social media.

This dissertation describes a new approach to knowledge acquisition and extraction that learns rich structures of events (e.g., plant, detonate, destroy) and participants (e.g., suspect, target, victim) over a large corpus of news articles, beginning from scratch and without human involvement. Early models in Natural Language Processing (NLP) relied on similar high-level representations like scripts (structured representations of events, their causal relationships, and their participants) and frames to drive interpretation of syntax and word use. Scripts, in particular, were central to research in the 1970s and 1980s for many applications. However, scripts were hand-coded and therefore unable to generalize to new domains. Modern statistical approaches and advances in NLP now enable new representations and large-scale learning over many domains.

This dissertation begins by describing a new model of events and entities called Narrative Event Schemas. A Narrative Event Schema is a collection of events that
occur together in the real world, linked by the typical entities involved. I describe
the representation itself, followed by a statistical learning algorithm that observes
chains of entities repeatedly connecting the same sets of events within documents.
The learning process extracts thousands of verbs within schemas from 14 years of
newspaper data. I present novel contributions in the field of temporal ordering to
build classifiers that order the events and infer likely schema orderings. I then present
several new evaluations for the extracted knowledge.

Finally, this dissertation applies Narrative Event Schemas to the field of Informa-
tion Extraction, learning templates of events with sets of semantic roles. A template
defines a specific type of event (e.g., a bombing) with a set of semantic roles (or
slots) for the typical entities involved in such an event (e.g., perpetrator, target, in-
strument). Most Information Extraction approaches assume foreknowledge of the
domain’s templates, but I instead start from scratch and learn schemas as templates,
and then extract the entities from text as in a standard extraction task. My algorithm
is the first to learn templates without human guidance, and its results approach those
of supervised algorithms.
Acknowledgments

This dissertation exists largely thanks to my advisor, Dan Jurafsky. I learned more from Dan than simply how to “do research”, but also how to communicate well, respect work from all universities, and find the positive side of negative results. I thank him for his guidance and support. I also thank the Stanford NLP group for rich conversations and a productive research environment, and to Chris Manning for his constant assistance and diligence in providing for the group. I particularly acknowledge collaborations both large and small from Marie-Catherine de Marneffe, Bill MacCartney, Dan Ramage, Steven Bethard, Heeyoung Lee, Andrey Gusev, Eric Yeh, Jenny Finkel, Mihai Surdeanu, David McClosky, and Ramesh Nallapati.

I also thank Andrew Ng and his Machine Learning course in which my first idea for this dissertation began, and thanks to Shan Wang for joining me in its first step.

This dissertation would not have started if not for the years of support from James Allen at the University of Rochester and IHMC. I am similarly in debt to my high school Computer Science teacher, Hal Honig. I credit where I am today to the mentoring and initial interest they both showed in me. They are engaged, caring educators, and the accomplishment that this dissertation represents is theirs as well.

Finally, my family is an unending source of support. To my parents, thank you for a 38 year example of what love looks like. To my brother, thank you for making me compete for everything in life. To my wife, Amy, words are too ambiguous to describe the unambiguous encouragement and unfailing support you give to me. I often wonder how many of these ideas would not be here without your feigned interest in them. To Eli, always expect great things. I thought of you while I wrote each of the following chapters.
## Contents

Abstract iv

Acknowledgments vi

1 Introduction 1

1.1 Learning Narrative Event Structure 3

1.1.1 Contributions 4

1.2 Learning the Temporal Ordering of Events 6

1.2.1 Contributions 7

1.3 Learning Events for Information Extraction 8

1.3.1 Contributions 9

1.4 Learning Knowledge Without Supervision 10

1.4.1 Contributions 12

1.5 Applications of Event Models 13

1.6 Layout of the Dissertation 16

2 Learning Narrative Event Chains 17

2.1 Comparison to Previous Work 19

2.2 The Narrative Chain Model 21

2.2.1 Definition 21

2.2.2 The Protagonist 22

2.2.3 Partial Ordering 24

2.3 Learning Narrative Relations 24

2.3.1 Evaluation Metric: Narrative Cloze 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.2 Narrative Cloze Experiment</td>
<td>27</td>
</tr>
<tr>
<td>2.4 Discrete Narrative Event Chains</td>
<td>29</td>
</tr>
<tr>
<td>2.5 Discussion</td>
<td>31</td>
</tr>
<tr>
<td>3 Learning Narrative Schemas</td>
<td>35</td>
</tr>
<tr>
<td>3.1 Narrative Schemas</td>
<td>38</td>
</tr>
<tr>
<td>3.1.1 Typed Narrative Chains</td>
<td>38</td>
</tr>
<tr>
<td>3.1.2 Learning Argument Types</td>
<td>39</td>
</tr>
<tr>
<td>3.1.3 Event Slot Similarity with Arguments</td>
<td>40</td>
</tr>
<tr>
<td>3.1.4 Narrative Schema: Multiple Chains</td>
<td>41</td>
</tr>
<tr>
<td>3.2 Sample Narrative Schemas</td>
<td>43</td>
</tr>
<tr>
<td>3.3 Evaluation: Frames and Roles</td>
<td>46</td>
</tr>
<tr>
<td>3.4 Evaluation: Coverage</td>
<td>51</td>
</tr>
<tr>
<td>3.4.1 Event Coverage</td>
<td>51</td>
</tr>
<tr>
<td>3.4.2 Results</td>
<td>53</td>
</tr>
<tr>
<td>3.5 Evaluation: Cloze</td>
<td>53</td>
</tr>
<tr>
<td>3.5.1 Narrative Cloze</td>
<td>54</td>
</tr>
<tr>
<td>3.5.2 Training and Test Data</td>
<td>54</td>
</tr>
<tr>
<td>3.5.3 Typed Chains</td>
<td>54</td>
</tr>
<tr>
<td>3.5.4 Narrative Schema</td>
<td>55</td>
</tr>
<tr>
<td>3.5.5 Typed Narrative Schema</td>
<td>55</td>
</tr>
<tr>
<td>3.6 Discussion</td>
<td>55</td>
</tr>
<tr>
<td>4 Learning to Order Events</td>
<td>59</td>
</tr>
<tr>
<td>4.1 Previous Work</td>
<td>60</td>
</tr>
<tr>
<td>4.2 Pairwise Classification</td>
<td>62</td>
</tr>
<tr>
<td>4.2.1 Data: The Timebank Corpus</td>
<td>63</td>
</tr>
<tr>
<td>4.2.2 Stage One: Learning Event Attributes</td>
<td>63</td>
</tr>
<tr>
<td>4.2.3 Stage Two: Event-Event Features</td>
<td>66</td>
</tr>
<tr>
<td>4.2.4 Evaluation and Results</td>
<td>68</td>
</tr>
<tr>
<td>4.2.5 Discussion</td>
<td>69</td>
</tr>
<tr>
<td>4.2.6 Pairwise Events Conclusion</td>
<td>70</td>
</tr>
</tbody>
</table>
7.2 Future Work ................................. 131
List of Tables

3.1 Five of the top fifty scored Narrative Schemas. Events and arguments in italics were marked misaligned by FrameNet definitions. * indicates verbs not in FrameNet. - indicates verb senses not in FameNet. . . . 45

4.1 Features selected for learning each temporal attribute. POS-2 is two tokens before the event. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65

4.2 Accuracy of the Stage One classifiers. . . . . . . . . . . . . . . . . . 65

4.3 Incremental accuracy by adding features. . . . . . . . . . . . . . . . 68

4.4 Top 5 features as added in feature selection w/ Naive Bayes, with their percentage improvement. . . . . . . . . . . . . . . . . . . . . . . . . . . 69

4.5 The features to learn temporal relations between two events. Asterisks (*) indicate features that are duplicated, one for each of the two events. 72

4.6 Two sets of confidence scores. The first set chooses before for all three labels, and the second chooses after. Other lower-scoring valid relation sets also exist, such as before, unknown, and before. . . . . . . . . . . . 73

4.7 The number of event-event relations after transitive closure. . . . . . 75

4.8 Using the base Timebank annotated tags for testing, accuracy on before/after tags in the two models. . . . . . . . . . . . . . . . . . . . . . . . . . . 76

4.9 The number of event-event before and after relations after transitive closure on each dataset. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81

4.10 Using the base Timebank annotated tags for testing, the increase in accuracy on before/after tags. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
4.11 Overall accuracy when training with different percentages of unknown relations included. 13% of unknowns is about equal to the number of befores. ................................................................. 84
4.12 Precision and Recall for the base pairwise decisions and the global constraints with integrated time information. .................................................. 85

5.1 Results for choosing the correct ordered chain. At least 10 indicates that there were at least 10 pairs of ordered events in the chain. .... 95

6.1 Coreference Vectors for the subjects of kidnap and release. The syntactic functions are counts of how many times each syntactic function contained an argument that coreferred with the main verb’s subject (e.g., kidnap or release). .................................................... 115
6.2 Selectional Preference vectors for the subjects of kidnap and release. .. 116
6.3 Document classification results on test. ........................................... 122
6.4 MUC-4 extraction, ignoring template type. ..................................... 124
6.5 Performance of the individual templates. The Attack column compares both of my 1 vs 5 best templates. ................................. 125
6.6 Performance of each template type, but only evaluated on the documents that are labeled with that template type. All others are removed from testing. The parentheses indicate F1 score gain over evaluating on all test documents (Table 6.5). ........................................ 125
List of Figures

1.1 A narrative schema: a typical output from my learning system. The variable notation ‘A search B’ indicates the syntactic subject and object positions that the variable fills. For example, A is the subject of the verb ’search’, and B is the object. ........................................ 3
1.2 The partial ordering for a set of learned events. .............................. 7
1.3 A template representing a Bombing Event in a newspaper article. ... 9

2.1 Two example narrative event chains. Each is a sequence of events and a parameterized entity that fills specific syntactic functions for each verb. The circles to the left of verbs represent the subject, and the right represents the object. Edges between circles indicate the same entity is constrained to fill those particular syntactic functions. ........ 18
2.2 Comparing two representations of a sentence's structure: a phrase structure tree and a typed dependency graph. ......................... 22
2.3 Three narrative events and the six most likely events to include in the same chain. ................................................................. 26
2.4 One of the 69 test documents, containing 10 narrative events. The protagonist is President Bush. ................................................... 28
2.5 Results with varying sizes of training data. Year 2003 is not explicitly shown because it has an unusually small number of documents compared to other years. ..................................................... 30
2.6 An automatically learned Prosecution Chain. Arrows indicate the before relation. ................................................................. 32
2.7 An Employment Chain. Dotted lines indicate incorrect before relations. 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>A narrative event chain and two as of yet unconnected events. Circles to the left of the verbs represent the syntactic subjects of the verbs, and the right represents objects. Edges indicate constraints between the syntactic functions that the entity must fill.</td>
<td>37</td>
</tr>
<tr>
<td>3.2</td>
<td>A typed narrative chain. The four top arguments are given. The ordering $O$ is not shown.</td>
<td>40</td>
</tr>
<tr>
<td>3.3</td>
<td>Merging typed chains into a single unordered Narrative Schema.</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>Graphical view of an unordered schema automatically built starting from the verb ‘arrest’. A $\beta$ value that encouraged splitting was used for this example. I ultimately set $\beta = 0.2$ by optimizing cloze performance on the development dataset.</td>
<td>44</td>
</tr>
<tr>
<td>3.5</td>
<td>Graphical view of an unordered schema automatically built from the verb ‘convict’. Each node shape is a chain in the schema.</td>
<td>46</td>
</tr>
<tr>
<td>3.6</td>
<td>Narrative schemas: examples were hand selected from the database to illustrate the diversity of learned narratives.</td>
<td>47</td>
</tr>
<tr>
<td>3.7</td>
<td>Narrative schemas: examples were hand selected from the database to illustrate the diversity of learned narratives.</td>
<td>48</td>
</tr>
<tr>
<td>3.8</td>
<td>Results on the narrative cloze test with varying sizes of training data. Both narrative chains and narrative schemas are compared with their untyped and typed versions.</td>
<td>56</td>
</tr>
<tr>
<td>4.1</td>
<td>Annotated relations in document wsj_0551.</td>
<td>77</td>
</tr>
<tr>
<td>4.2</td>
<td>The straightforward procedure that compares two time intervals.</td>
<td>80</td>
</tr>
<tr>
<td>4.3</td>
<td>Before and after time-time links with closure.</td>
<td>83</td>
</tr>
<tr>
<td>5.1</td>
<td>Four passages from the NYT section of the Gigaword Corpus containing arrest and plead. The temporal classifier classifies each passage independently, determining if each pair is in a before or other relation. The second example illustrates a difficult case involving multiple arrest events.</td>
<td>91</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Before and after counts between pairs of events in the criminal domain: counts of classification decision over the NYT portion of the Gigaword Corpus.</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>A narrative chain and its fully reversed partial ordering.</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>Passages from the NYT where <em>acquit</em> was classified as before <em>convict</em>. All three are correct classifications involving different trial periods. Two of the three are hypothetical convictions described with modalities occurring in the future.</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Passages from the NYT where <em>acquit</em> was classified as before <em>convict</em>. These examples illustrate that sentence-internal ordering of past tense verbs favors before relations.</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>Examples of text and the gold labeled BEFORE relations in the Time-bank Corpus for the event <em>retire</em>.</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>Examples of obituaries where the <em>retire</em> event is incorrectly classified as after the <em>die</em> event.</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>A template from the MUC-4 corpus.</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>A portion of a MUC-4 document with no templates to extract.</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>The 4 clusters mapped to MUC-4 templates.</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>Three learned example templates representing three of the original MUC-3 hand-coded templates.</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>Two learned example templates that were not hand-created in the MUC-3 corpus. All knowledge except the template/role names (e.g., ‘Victim’) is learned.</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>Slots in the hand-crafted MUC-4 templates.</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The early years of Natural Language Processing (NLP) took a “top-down” approach to language understanding by using high-level representations like scripts (structured representations of events, their causal relationships, and their participants) and frames to drive interpretation of syntax and word use. Scripts, in particular, were central to research in the 1970s and 1980s for proposed tasks such as summarization, coreference resolution and question answering. For example, Schank and Abelson (1977) proposed that understanding text about restaurants requires characteristic knowledge about a Restaurant Script. The Restaurant Script included a stereotypical restaurant’s participants (Customer, Waiter, Cook, Tables, etc.), the events constituting a visit to a restaurant (entering, sitting down, asking for menus, etc.), and the various preconditions, ordering, and results of each of the typical actions that occur. They argued that knowledge structures such as these provide a language understanding and reasoning system (including the human mind) with rich information about many aspects of meaning.

However, the problem with scripts and similar common-sense knowledge structures was that the need for hand construction, specificity, and domain dependence prevented robust and flexible language understanding. The many diverse and varied situations in the world cannot realistically be hand coded for every language application and domain. For example, the Restaurant Script cannot assist the understanding situations like corporate acquisitions and football games. The development of scripts
proved too time intensive and too brittle when changing contexts.

In contrast, modern work on language understanding offers statistical techniques to learn diverse knowledge automatically from free text. Machine learning typically focuses on shallower representations of meaning than scripts. Semantic roles, for instance, express at least one aspect of the semantics of events and have proved amenable to supervised learning from annotated corpora like PropBank (Palmer et al., 2005) and Framenet (Baker et al., 1998). Creating these supervised corpora is an expensive and difficult multi-year effort, requiring complex decisions about the exact set of roles to be learned. While these supervised approaches are thus more robust than scripts, they also share similar difficulties in regards to the amount of human effort and annotations that are required. Even unsupervised attempts to learn semantic roles often require a pre-defined set of roles (Grenager and Manning, 2006a) or a hand-labeled seed corpus (Swier and Stevenson, 2004a; He and Gildea, 2006a). Further, their shallow nature does not capture important interactions between entities and events that are needed for full document understanding.

This dissertation describes how to integrate the strengths of hand-created event representations with those of today’s learning algorithms for shallow semantics. I present the first approach to learning rich common-sense knowledge about events in the world in the form of narrative event schemas, but without pre-defined frames, roles, or tagged corpora. The learning algorithm induces both event structures and the roles of their participants from millions of unlabeled newspaper articles. I introduce a new indicator of similarity, the protagonist, and show how discourse-level connections between predicates can direct the learning process. Further, I describe a method of temporal reasoning that learns to order the learned events, providing a new approach to temporal reasoning that is central to theories of causation and inference. I conclude by showing how this learning approach can be applied to a common information extraction application in which all previous work has depended on hand-coded definitions of event structure. This algorithm is the first to perform structured extraction without knowing the event structure in advance, and I present results that approach the performance of these knowledge-dependent algorithms.
Figure 1.1: A narrative schema: a typical output from my learning system. The variable notation ‘A search B’ indicates the syntactic subject and object positions that the variable fills. For example, A is the subject of the verb ’search’, and B is the object.

1.1 Learning Narrative Event Structure

The central focus of this dissertation is learning stereotypical sets of events and entities that define common-sense situations in the world. Figure 1.1 graphically illustrates one such Narrative Event Schema about a criminal prosecution, to be defined more formally later. This example was learned completely automatically from a corpus of newspaper articles. A narrative event schema is a set of related events (left), a set of participants (right), and constraints on the syntactic positions that the participants fill (variables on left with the verbs, indicating their subject and object positions). All three aspects of a schema are jointly learned from unlabeled text, including the ordering of the events. The six events on the left follow four participants on the right through a set of events that constitute a narrative.

Robustly learning these schemas could assist a variety of NLP applications, most significantly in applications that require a document-level understanding of event descriptions as they unfold across sentence boundaries, and the roles that entities play in the overall narrative. The document summarization task, for example, condenses a document by identifying which sentences describe the story’s main events. Identifying those events and filtering out tangential knowledge is one of the key challenges of summarization. Learning sets of related events across a diverse array of topics, such as those in narrative event schemas, could assist in identifying the main events and
robustly solving this task. Similarly, question answering and information extraction applications require algorithms that can identify the central entities and what roles they play in a document (e.g., the perpetrator and victim of a crime). Such algorithms need to be seeded with knowledge about how different entity types interact with different events. Discovering and learning this knowledge is a central challenge to solving both applications. Finally, event prediction and anomaly detection need knowledge of stereotypical sets of events in the world in order to differentiate between expected and unexpected occurrences. Narrative schemas provide the representation to address all of these tasks, and this dissertation describes how to learn and extract such structured schemas from unlabeled newspaper articles.

1.1.1 Contributions

My work on narrative schemas makes several key contributions to the fields of event semantics and representation, learning events without human supervision, and semantic role modeling.

The main contribution is the narrative event schema representation itself. As opposed to my joint model of both events and entities, previous work on modeling events and topics focused on bag-of-words models (e.g., arrest, raid, and capture) to characterize a domain. This includes work in the areas of topic modeling (Blei et al., 2003; Lin and Hovy, 2000) and event clustering (Bejan, 2008), as well as approaches that focus on the text order itself to model event sequences (Fujiki et al., 2003; Manshadi et al., 2008; Gordon, 2010). Most of these approaches only learn bags of event words. My contribution to this area is to learn entities with the events, and to define their functions within the event words. Preliminary work in learning richer structure includes toy domains (Mooney and DeJong, 1985) and synonymous relations (Brody, 2007). This dissertation is the first to use modern statistical techniques to learn rich narrative structure, jointly learning the events, their key participants, and the roles of the participants without human supervision from open-domain text. The narrative schema is thus a new knowledge representation, bridging the early interest in Schankian scripts to today’s statistical learning techniques. Recent work since the
initial publication of this representation shows usefulness in other areas as well, such as story analysis and generation (McIntyre and Lapata, 2009).

Another important contribution is the protagonist-based learning algorithm and its applications to the broader field of word similarity and distributional learning. Word similarity judgments are important to a wide range of NLP applications, including search, parsing, machine translation, and lexical semantics. Most approaches to measuring similarity rely on modeling the context in which words typically appear. Broadly, this is accomplished by counting words that tend to occur with each target word. Two words are judged to be similar if similar words occur in both contexts. My contribution to the field is a new discourse-level approach that looks at the interactions between words, using discourse-level relations, rather than bag-of-word contexts. My algorithm focuses on entity mentions in text and uses repeated mentions of the same entity to identify related events. For instance, the phrases ‘Bruce pushed the man’ and ‘the man fell down’ convey implicit knowledge about the world, namely, that push and fall down are related. We know this because the man appears with both verbs. We can thus loosely conclude that the two events occur together in the real world. The man plays the role of the protagonist in this narrative, and two verbs are deemed similar if they occur together in documents with the same entity. This is a significant departure from current similarity measures in that I use discourse-level information to decide relatedness. Chapter 2 describes the algorithm in detail.

The final contribution of narrative schemas is a new perspective on semantic role labeling. As described above, the protagonist learns to connect verbs like push and fall, but it also learns a more specific connection: the object of pushed (the patient) is the same entity as the subject of falls (the theme). Traditionally, these individual verb positions are called semantic roles, defining the function (e.g., patient) that an argument (e.g., the man) fills for a verb (e.g., push). Several corpora exist for semantic roles (Palmer et al., 2005; Baker et al., 1998), and lots of work focuses on learning to label syntactic arguments with their roles (Gildea and Jurafsky, 2002; Grenager and Manning, 2006a). My work on narrative schemas does not learn these verb-specific roles, but rather the broader situation-specific roles. In a narrative about criminal activity, the man is not the patient or theme of the overall activity as it is
of individual verbs, but instead fills a broader victim role across all verbs. These broader roles are useful for applications like information extraction, as is described in chapter 6. Recent work since the publication of the protagonist has even showed that verb-specific semantic roles can benefit from this broader view (Gerber and Chai, 2010). Defining situation-level semantic roles is a largely unexplored area in NLP, and my work presents one possible method of learning and representing them.

Narrative event schemas thus offer contributions in three main areas: jointly representing and learning events and entities, a protagonist-based approach to word similarity, and a situation-level view of semantic roles.

1.2 Learning the Temporal Ordering of Events

After learning a representation of events and entities, the next task is to impose a real-world ordering over the events. The goal is to learn that events like arrest occur before convict. Figure 1.2 graphically illustrates the structure of a narrative event schema before and after such a learned ordering is applied. This criminal prosecution ordering was fully learned from a corpus of newspaper articles. In order to learn this ordering, I use the Timebank Corpus to train supervised learning models that classify temporal relations between events (e.g., before, simultaneous, includes, etc.). A significant aspect of this dissertation presents two approaches to recovering the temporal order of events, both a pairwise local model (independent local decisions), and a global model that enforces consistency over transitivity relations. These advances in time classification are then applied to learn and impose a partial ordering over narrative event schemas.

Being able to temporally order events is a necessary component for complete document understanding. In fact, extracting the temporal information in text descriptions of events has been of interest since the early days of natural language processing. Prediction and inference tasks, such as inferring other events likely to occur after an observed sequence of events, cannot be solved without a model of time. Lately, it has seen renewed interest as question answering, information extraction and summarization applications find it critical in order to proceed beyond surface understanding.
Research in causation is also very important to deep reasoning systems that rely on logical inference and more strict measures of entailment. While my goal is not to explicitly learn causation, learning event orderings could be used to approximate and assist in decisions of causation.

1.2.1 Contributions

This dissertation makes two main contributions to the field of temporal reasoning and learning event orders: (1) a new state-of-the-art pairwise classifier, and (2) the first global model to enforce ordering consistency.

The first contribution is the improvement of supervised learning models to classify pairs of events (e.g., is arrest before or after convict?). The creation of the Timebank Corpus (Pustejovsky et al., 2003) facilitated the development of supervised learning techniques for event ordering applications. Timebank labels event words and temporal relations between pairs of events that reflect their real-world order. Early research on Timebank focused on the event-event ordering task, but used gold information in the annotated corpus as features (e.g., verb tense, aspect, polarity) (Mani et al., 2007). My work was the first to solve the task completely automatically, predicting attributes like tense and aspect, rather than using gold labels. I also expanded the feature set to use deeper syntax and discourse connections to produce a state-of-the-art classifier. Since then, the TempEval contests (Verhagen et al., 2007, 2009) have encouraged continued work in the area. Most approaches still focus on pairwise event-event decisions (Hagege and Tannier, 2007; Bethard and Martin, 2007; Cheng
et al., 2007). One drawback of these models is that they often make local decisions that are globally inconsistent.

My second contribution to the field addresses this inconsistency in pairwise classifiers. I developed the first global framework for event-event ordering that informs local decisions (e.g., \( A \) before \( B \)) with document-level constraints (e.g., transitivity rules). I use two types of implicit global constraints: transitivity (\( A \) before \( B \) and \( B \) before \( C \) implies \( A \) before \( C \)) and time expression normalization (e.g., \( \text{last month} \) is before \( \text{yesterday} \)). I show how these constraints can be used to create a more densely-connected network of events, and how global consistency is enforced by incorporating these constraints into an integer linear programming framework. My approach builds on related work in paragraph ordering from Bramsen et al. (2006). I present results in chapter 4 on two event ordering tasks that show significant increases over a pairwise model. Since then, others have developed complementary global models based on my approach (Yoshikawa et al., 2009).

### 1.3 Learning Events for Information Extraction

Information Extraction (IE) refers to a range of applications that attempt to identify (and extract) information in text. Most algorithms focus on extracting binary relations and atomic facts, such as authors and book titles, capitals and countries, actors and birthdays, etc. Both supervised and unsupervised learning algorithms have proved capable of extracting these types of relations. This dissertation, however, focuses on a more structured type of extraction often called template-based extraction. Template-based IE attempts to capture an entire situation’s context, rather than disconnected facts and relations. For example, figure 1.3 shows a bombing template that defines several entity types, including a \textit{perpetrator}, \textit{target}, \textit{victim}, and an \textit{instrument}. For any given document, can we identify its overall theme, extract its central entities, and identify the template roles the entities play in it? Template-based IE is concerned with filling in the values of these roles from a document. I learned this example’s template structure from raw unlabeled text.

Information extraction algorithms are important to improving many user-end
tasks such as search and retrieval, database population, and language understanding. Search and retrieval are the most obvious applications. Whereas most current retrieval applications rely on returning entire documents, IE is relevant to more fine-grained search like snippet generation and question answering applications. In contrast, database population applications map textual descriptions into queriable databases that can be more easily organized and searched. For instance, extracting business names and phone numbers from webpages into a database provides a more accessible interface to this knowledge. Finally, template-based IE shows promise for many areas of language understanding. Identifying the template roles of entities can assist summarization, multi-document clustering, and related document understanding applications.

1.3.1 Contributions

This dissertation is the first approach to template-based IE that automatically learns template structure from unlabeled data before extracting its answers. Standard algorithms for template-based IE require predefined template schemas (Chinchor et al., 1993; Rau et al., 1992), and often labeled data (Freitag, 1998; Chieu et al., 2003; Bunescu and Mooney, 2004; Patwardhan and Riloff, 2009), to learn to extract their slot fillers (e.g., an embassy is the Target of a Bombing template). My approach begins without labeled data and without the template schemas, effectively not knowing what type of events are described in the text. By instead learning script-like knowledge automatically, I remove the knowledge requirement and perform extraction without knowing the structure in advance. My algorithm instead learns automatically from

<table>
<thead>
<tr>
<th>Instantiated Bombing Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perpetrator: 'The Farabundo Marti Liberation Front'</td>
</tr>
<tr>
<td>Target: 'Mayor’s Office'</td>
</tr>
<tr>
<td>Victims: 'the gate guard'</td>
</tr>
<tr>
<td>Instrument: 'ten sticks of dynamite'</td>
</tr>
</tbody>
</table>

Figure 1.3: A template representing a Bombing Event in a newspaper article.
raw text, learning template schemas as sets of linked events (e.g., bombings include *detonate*, *set off*, and *destroy* events) associated with semantic roles (e.g., bombings include *perpetrators* and *victims*). Once induced, I extract role fillers from specific documents using learned syntactic patterns. Most recently, unsupervised IE algorithms focus on learning binary relations for atomic fact extraction (Banko et al., 2007b; Carlson et al., 2010b,a; Huang and Riloff, 2010; Durme and Pasca, 2008a), but my work is unique in focusing on full templates with their interconnected entities and events. See the next section for more on this distinction. I evaluate on a common extraction dataset and show that I induce template structure very similar to hand-created gold structure, and I extract role fillers with performance approaching algorithms requiring knowledge of the templates.

1.4 Learning Knowledge Without Supervision

The learning algorithms in this dissertation are mostly unsupervised, requiring no human annotators, sorting of text, and no seeds from which to bootstrap schemas\(^1\). Unsupervised and semi-supervised knowledge extraction has received new attention in recent years, and this dissertation is no exception to the trend. Learning from raw text carries the benefit of removing dependencies on human annotators, and allowing a system to immediately learn within new domains and languages. Unsupervised knowledge acquisition typically falls into one of three types of target knowledge: (1) ontology induction, (2) attribute extraction, and (3) fact extraction and relation learning. I briefly describe each here, and then present the contributions of this dissertation.

Ontology induction extracts *is-a* relations from text to build a hierarchy of types (e.g., cat *is-a* mammal). Traditionally, seed-based learning is used to begin with a few hand-built patterns (e.g., *X is a type of Y*), and then bootstrap to identify new text patterns for the relation. A few examples of paths through an automatically

\(^1\)To be clear, I do rely on NLP tools that are supervised, such as an English parser and a named entity recognizer. These are trained on English syntax and named entity labels. I thus do not claim to be fully unsupervised.
learned ontology are shown here.

- dog is-a mammal is-a animal is-a living-thing
- car is-a vehicle is-a transport is-a object
- microsoft is-a company is-a organization

A range of semi-supervised bootstrapping, as well as unsupervised approaches, have been used to learn ontologies (Hearst, 1992; Durme and Pasca, 2008a; Poon and Domingos, 2010). Most recently, this is now coupled with attribute extraction to improve performance of concept clustering and the richness of extracted knowledge.

Attribute extraction focuses on learning common attributes of objects. The target objects range from people and animals to consumer products like ipods and cameras. The following are learned examples from Pasca (2008).

- sea creatures have habitats, reproduction, and food chains
- painters have artwork, self portraits, and biographies
- physicists have inventions, biographies, and techniques

Pasca (2008) and others use seed-based and weakly supervised approaches to jointly learn these concept attributes and ontology structure (Pasca, 2008; Reisinger and Pasca, 2009). The result is an ontology of concepts and hierarchical attributes. Attribute learning has been applied to different types of data, ranging from web query logs (Pasca and Durme, 2007; Durme and Pasca, 2008b), open-domain text (Yoshinaga and Torisawa, 2007), and Wikipedia articles (Suchanek et al., 2007; Wu and Weld, 2008).

Finally, fact and relation extraction focuses on extracting true statements about the world. These are typically in the form of relation triples: A relation B. Examples from the Open IE work of Banko et al. (2007b) and Banko (2009) are given here:

- Napoleon married Josephine
- Einstein born in Ulm
- oranges contain Vitamin C
- XYZ Corp. acquired Go Inc.
Open IE is largely unsupervised, and can be viewed as learning relations that are true about the world from a large corpus of text. It is important to note that the system does not seek to understand any given document, but depends on redundancy across thousands of documents to extract relations. The system learns synonymous patterns (e.g., become member of, enter, and join), as well as classes of similar concepts (e.g., company, business, inc., organization) by observing patterns with the same argument types. Approaches vary in their use of (or lack of) seed examples, domain text, and target knowledge (Kok and Domingos, 2008; Carlson et al., 2010b,a; Huang and Riloff, 2010).

1.4.1 Contributions

This dissertation learns a complimentary, but different type of knowledge from the above work on relation learning. The type of knowledge represented by concept ontologies and their attributes is not addressed in this dissertation. The knowledge output by systems like Open IE (Banko et al., 2007b) and the Never-Ending Language Learning architecture of (Carlson et al., 2010a) are lists of independent relations. The schemas in this dissertation represent sets of dependent relations and entities in specific roles. This dissertation is thus unique in learning relations across relations, or imposing a structure over otherwise independent relations. My relations are intentionally focused on a narrative semantics, capturing relations that represent events occurring together in the world.

One key difference in my algorithm is that I do not capture specific facts, like napoleon married josephine, but instead capture general events/relations like People marry People. Factoid learning systems could also learn People marry People, but they do not connect marry to other related events, such as pregnant, loves, and buys a house. This level of semantics and richer connections between entities and events has not been addressed in current relation learning systems.

Finally, this dissertation also differs by performing a document extraction task that other relation learning systems cannot perform. As I discussed above, relation learning learns things that are true, but does not understand any particular single
document. They are essentially large-scale fact finders. For instance, Open IE only extracts facts and patterns that are seen repeated over many documents. Chapter 6 presents an information extraction task that requires knowing and extracting both the perpetrator and target of a crime from a single document, perhaps each only mentioned once in the corpus. Further, independent relations cannot make connections across events to link perpetrators, victims, targets, and other entities. Schemas, on the other hand, do represent these entities in one structure. Narrative schemas offer a joint model of events and entities that help us perform this level of document understanding and extraction.

1.5 Applications of Event Models

Just as scripts motivated many natural language applications in their time, narrative schemas have similar connections to the problems addressed in many modern applications of language understanding systems. The type of application that is best suited for knowledge about entities and events is that which deals with an entire document’s context. Sentence-based and lower-level language tasks, like parsing and named entity recognition, have less of a need for this type of knowledge as they typically focus on a more limited context that doesn’t rise to the level of events. This section briefly looks at the higher-level applications in which narrative schemas and similar event-based models may assist.

**Summarization:** Summarization is the task of condensing a document into a shorter form while maintaining the key ideas and facts contained therein. The vast majority of work in this field is *extractive summarization*: identify the most important sentences and concatenate them into a summary. This is often conducted with multiple news stories about the same situation, and redundancy across the documents is used as an indicator of importance. Other features like position in the document are important, but redundancy remains the most critical. Narrative schemas could provide a crucial bit of semantic information by identifying which event words are
most probable in a previously learned scenario, thus providing a new piece of information that is otherwise unavailable to current approaches. Several approaches suggest such knowledge may have merit. For instance, focusing on key entity mentions and coreference chains can better guide sentence selection (Azzam et al., 1999; Bergler et al., 2003). Most notably for this dissertation, Filatova and Hatzivassiloglou (2004) showed that focusing on event words is often more useful than the entire sentence’s context. Models, such as narrative schemas, that include both events and entities may further inform the task.

Coreference Resolution: Coreference resolution is the linking of entity mentions that refer to the same entity. For instance, Nate Chambers and Chambers both refer to myself, as well as the pronouns he and him in the correct context. The task requires linking these mentions throughout a document. New models of a document’s events and how its participants interact have obvious applications to this task. Bean and Riloff (2004) applied a preliminary idea related to schemas, called caseframes, to coreference resolution. They counted pairs of verbs whose arguments were exact string matches (e.g. Reid ran and found Reid) in a domain-specific corpus, and used those observed pairs to generalize that two argument positions were related (e.g. the subject of ran and the object of found). They showed a relative improvement in coreference performance. This result suggests that narrative schemas, which encode entire sets of events, may provide added benefit.

Event Reasoning and Inference: Logical inference and deeper reasoning require temporal and causal relations to identify semantic connections between a document’s sentences. Early work on Schankian scripts frequently referenced Reasoning and Inference as the driving motivation for their work. Even today, most work on models of events and common-sense knowledge make reference to reasoning systems as a possible application. Event structures like narrative schemas lend themselves quite naturally to inference because they explicitly define an event ordering. This ordering can explicitly or implicitly be used to infer missing or future events. Paul et al. (2009) has recently used coreference patterns to learn reciprocal relationships. For
instance, a user reading a document that contains the events *arrest* and *sentence* may wish to infer what other events had occurred. A narrative schema that contains the event chain, *arrest - plead - convict - sentence*, can infer that *convict* and *plead* likely occurred as well. These types of inferences are most helpful in deep reasoning systems that need to understand events beyond the surface information described by a document. Schemas are not the complete solution, but are a step toward learning these important relations.

**Question Answering:** Question Answering systems seek specific responses that address a targeted query in the form of a question. As opposed to information retrieval where the goal is a list of ranked documents that are relevant to a query, question answering systems return short strings that answer the query’s question. For example, *Who is the suspect in yesterday’s embassy bombing?*, might return a single answer, *John Doe*. Most algorithms rely on reformulating the question into a statement, and then search a large corpus for text that matches the statement (e.g., *John Doe is the suspect of the bombing*). Unfortunately, the answer is often not explicitly given in such a straightforward reformulation. Narrative schemas could help solve this problem through its representation of entities and their roles in events. By representing a *suspect* with a series of events, schemas describe alternate ways that language might describe the suspect beyond the question’s particular word choices.

**Information Extraction:** Information Extraction (IE) is the task of extracting particular pieces of knowledge from text. This includes large-scale web search where the information is described in an as yet unknown document, or more specific search where the search is within a specific document from which the information must be extracted. Many IE applications begin with the desired type of information (e.g., the capitals of U.S. states), and the task is to extract a database of results (e.g., Richmond/Virginia, Annapolis/Maryland, Springfield/Illinois, etc.). However, recent work focuses on learning what type of information exists in the world, and automatically creates knowledge bases of unknown facts. As described above in Section 1.3, narrative schemas offer a novel approach to representing events and situations in the
real world, and further, to assist in their extraction from written language. This dissertation focuses on the last application, Information Extraction, in Section 6 to evaluate and ultimately show the utility of learning narrative schemas from open-domain, unannotated text.

1.6 Layout of the Dissertation

This dissertation begins by describing how to learn unordered Narrative Schemas from large amounts of text, followed by my advances in the supervised learning of temporal relations between events, an implementation of ordered Narrative Schemas, and finally concluding with a template-based information extraction application.

Chapter 2 lays the underlying framework for learning Narrative Schemas. It introduces the protagonist as the key intuition to discovering event relations, and shows how to learn Narrative Chains, sets of related events connected by a single entity. Chapter 3 then expands this narrative chain framework and describes a new learning algorithm that leverages the protagonist to jointly learn events and all participating entities. My contributions to event representations, the novel learning algorithm based on a protagonist, and semantic roles for narratives are found in these two chapters.

Chapter 4 shifts gears and focuses on the event ordering task. This chapter describes my novel supervised learning approach to ordering events, as well as my contribution of maintaining global consistency across time relations within an Integer Linear Programming framework. Chapter 5 then describes how to use a supervised classifier to order generalized learned narrative event schemas.

Finally, chapter 6 describes my contribution to template-based information extraction: it is the first approach to automatically learn templates on a common evaluation dataset, and then compares my extraction performance against supervised learning systems on the same task. I show that I can extract with performance approaching the results of other approaches that assumed full knowledge of the domain and access to gold-labeled data.
Chapter 2

Learning Narrative Event Chains

The central focus of this dissertation is learning narrative event schemas to represent common situations by jointly modeling events and entities, and the constraints between them. However, before I learn entire schemas from raw text, this chapter introduces a partial representation of events called narrative event chains (or narrative chains) that will form the groundwork for learning complete schemas. A narrative chain is a simplification of narrative schemas that follows a single entity independent of other entities that are involved in the events. This single entity, called the protagonist, forms the basis of the narrative learning process. Rather than learning all entities together, a chain learns one entity and its sequence of related events. This chapter formalizes the narrative chain, describes its protagonist-based learning algorithm, and evaluates its learned knowledge on an event prediction task. The following chapter will then expand this learning process to learn full narrative schemas.

Narrative chains are partially ordered sets of events centered around a common entity, called the protagonist. As described in Chapter 1, they are related to the structured sequences of participants and events that have been called scripts (Schank and Abelson, 1977) or Fillmorean frames. However, chains instead focus on a single participant in each event sequence. This participant and its events can be filled in and instantiated for a particular document to draw inferences. Such inferences are important to many applications, including but not limited to, summarization, information extraction, and coreference resolution.
Consider the two distinct narrative chains in figure 2.1. It would be useful for question answering or textual entailment to know that ‘someone denied’ is also a likely event in the left chain, while ‘someone replaces’ temporally follows the right. Narrative chains (such as Firing of Employee or Executive Resigns) offer the structure and power to directly infer these new subevents by providing critical background knowledge. In part due to its complexity, automatic induction of event models such as these has not been addressed since the early non-statistical work of Mooney and DeJong (1985).

The first step to narrative induction is an entity-based approach to learning related events that follow a protagonist. Using a protagonist to direct the learning process is one of the main contributions of this dissertation. The model is inspired by Centering (Grosz et al., 1995) and other entity-based models of coherence (Barzilay and Lapata, 2005). As a narrative progresses through a series of events, each event is characterized by the grammatical role played by the protagonist, and by the protagonist’s shared connection to surrounding events. My algorithm is an unsupervised distributional learning approach that uses coreferring entity mentions as evidence of a narrative relation between two events. I use the New York Times section of the Gigaword Corpus, Third Edition (Graff, 2002) for learning, observing coreferring entity mentions within one million newspaper articles. The result is a large, diverse...
set of narrative chains that represent a wide range of domains. I also show, using a new evaluation task called narrative cloze, that the protagonist-based learning approach leads to better induction than a verb-only distributional approach.

After learning a chain of events, the next step is to order the events. Chapter 5 describes how to apply work in the area of temporal classification to create partial orders of the learned events. Finally, I conclude this chapter by showing how the space of narrative events can be clustered and pruned to create discrete sets of narrative chains.

2.1 Comparison to Previous Work

*Topic signatures* and *topic models* are common ways to model word usage within a particular topic or situation. As opposed to chains, they don’t contain structure, but are based on bag-of-words representations. Topic signatures are extracted from hand-sorted (by topic) sets of documents using log-likelihood ratios (Lin and Hovy, 2000). The likelihood ratio is used to determine if two words occurred more often than chance within a set of documents. Topic models, such as Latent Dirichlet Allocation, are probabilistic models that treat documents as mixtures of topics. They learn topics as discrete distributions (multinomials) over words (Blei et al., 2003). Both approaches can capture event words and some narrative relations, but they lack any representation of entity and event interactions. As will be seen, my results illustrate the amount of noise present when you ignore entities and rely purely on co-occurrence information.

Mooney and DeJong (1985) is one of the earliest works on learning Schankian scripts from text examples. They used a small set of simplified English sentences that describe stories to learn generalized schemas with the goal of a question-answering system based on this knowledge. More recently, Bean and Riloff (2004) proposed the use of caseframe networks as a kind of contextual role knowledge for anaphora resolution. A caseframe is a verb and a semantic role (e.g., *<patient> kidnapped*). Caseframe networks are relations between caseframes that may represent synonymy.
(<patient> kidnapped and <patient> abducted) or related events (<patient> kidnapped and <patient> released). Bean and Riloff learn these networks from two topic-specific texts and apply them to the problem of anaphora resolution. My work can be seen as an attempt to generalize the intuition of caseframes (finding an entire set of events rather than just pairs of related caseframes) and apply it to a different task (finding a coherent structured narrative in non-topic-specific text).

Brody (2007) also proposed an approach similar to caseframe networks that discovers high-level relatedness between verbs by grouping verbs that share the same lexical items in subject/object positions. He calls these shared arguments anchors. He learned pairs of related verbs, similar to the results with caseframes. A human evaluation of these pairs shows an improvement over baseline. This and previous caseframe work lend credence to learning relations from verbs with common arguments.

I also draw intuition from lexical chains (Morris and Hirst, 1991; Barzilay and Elhadad, 1997), indicators of text coherence from word overlap/similarity. They found that repeated word usage correlates to overall text coherence. Barzilay and Lapata (2005) use a similar technique to improve coherence in a summarization application. My use of a repeated protagonist across verb arguments can be viewed as a type of lexical chain. To the best of my knowledge, this approach is the first to apply these assumptions to acquire explicit knowledge from text, rather than simply model text coherence.

Work on semantic similarity learning such as Chklovski and Pantel (2004) also automatically learns relations between verbs. They used hand-coded query patterns like “X and then Y” to retrieve matches from a search engine, and collected pairs of words that filled the pattern’s parameters. My work does not rely on specific patterns or hand-coded knowledge, but I utilize their mutual information distributional scoring metric between verbs. I also differ with my use of a protagonist as the indicator of relatedness, and I learn richer structure by learning how entities interact with my learned events.

Fujiki et al. (2003) investigated script acquisition on a limited domain by extracting the 41 most frequent pairs of events from the first paragraph of newspaper
articles, using the assumption that the paragraph’s textual order follows temporal order. Manshadi et al. (2008) and Gordon (2010) built n-gram models of verb-object pairs from “story genre” blogs as a way of modeling event sequences. They show performance of 64.7% in predicting a document’s event order, but the connection between blog order and real-world order has not yet been studied. Bejan (2008) also modeled event scenarios by utilizing Latent Dirichlet Allocation to cluster event words in the Timebank Corpus (Pustejovsky et al., 2003). Finally, Bejan (2009) looked into semantic roles of events, using supervised learning of specific verbs based on labeled FrameNet data.

In contrast to all previous work on modeling events, I learn event structures for thousands of events, jointly induce the structures, and learn who and how the participants are involved. Further, my learning algorithm does not depend on annotated corpora, simplified text, or other human intervention.

Since initial publication of my results, Regneri et al. (2010) looked into crowdsourcing schema creation, and Kasch and Oates (2010) showed how web queries can alternatively inform schema learning.

### 2.2 The Narrative Chain Model

#### 2.2.1 Definition

This model is inspired by Centering (Grosz et al., 1995) and other entity-based models of coherence (Barzilay and Lapata, 2005) in which an entity is in focus through a sequence of sentences. I use this same intuition to induce narrative chains. I assume that although a narrative has several participants, there is a central actor who characterizes a narrative chain: the protagonist. Narrative chains are thus structured by the protagonist’s grammatical roles in the events.

The task, therefore, is to learn events that constitute narrative chains. Formally, a narrative chain is a partially ordered set of narrative events that share a common actor. A narrative event is a tuple of an event (most simply a verb) and its participants, represented as typed dependencies. Typed dependencies represent the
grammatical relationships between tokens in a sentence (e.g., *the dog* is the *subject* of *barks*). This is in contrast to the syntactic phrase structure representation of sentences. Figure 2.2 graphically compares the two representations. Typed dependencies naturally fit event structures that are more concerned with higher level relations (e.g., subjects, objects) rather than lower level phrase structure (e.g., VP – VB NP). Since I am focusing on a single actor in this study, a narrative event is thus a tuple of the event and the typed dependency of the protagonist: \( \langle \text{event}, \text{dependency} \rangle \). An ordered narrative chain is a set of narrative events \( \{e_1, e_2, \ldots, e_n\} \), where \( n \) is the size of the chain, and a relation \( B(e_i, e_j) \) that is true if narrative event \( e_i \) occurs strictly before \( e_j \) in time. I will address the question of time in chapter 5.

### 2.2.2 The Protagonist

The notion of a protagonist motivates my approach to narrative learning. I make the following assumption of narrative coherence to drive the learning process.
CHAPTER 2. LEARNING NARRATIVE EVENT CHAINS

Narrative Coherence Assumption

Verbs that share coreferring arguments within a document are semantically connected by virtue of narrative discourse structure.

A single document may contain more than one narrative (or topic), but the narrative coherence assumption states that a series of argument-sharing verbs is more likely to participate in a narrative chain than those not sharing. In addition, this narrative approach captures grammatical constraints on narrative coherence. Simple distributional learning might discover that the verb *push* is related to the verb *fall*, but narrative learning can capture additional facts about the participants, specifically, that the object or patient of the *push* is the subject or agent of the *fall*.

Each focused protagonist chain offers one perspective on a narrative, similar to the multiple perspectives on a commercial transaction event offered by buy and sell. The following passage from a newspaper article illustrates how a protagonist connects related events.

*The oil stopped gushing from BP’s ruptured well in the Gulf of Mexico when it was capped on July 15 and engineers have since been working to permanently plug it. The damaged Macondo well has spewed about 4.9m barrels of oil into the gulf after an explosion on April 20 aboard the Deepwater Horizon rig which killed 11 people. BP said on Monday that its costs for stopping and cleaning up the spill had risen to $6.1bn.*

In bold font are four entity mentions (BP’s ruptured well, it, it, damaged Macondo well) of the protagonist in this paragraph, linked through coreference resolution. These four entity mentions connect four different event words (gushing, capped, plug, and spewed). All four events are highly related to each other, and the protagonist automatically connects them for the learning algorithm. Other less-related events (e.g., working, said, and risen) are not connected. Previous work that relies on simple co-occurrence statistics cannot differentiate between these two sets as the protagonist does. The next sections more formally describe and evaluate this intuition.
2.2.3 Partial Ordering

An ordered narrative chain, by definition, includes a partial ordering of events. Early work on scripts included ordering constraints with more complex preconditions and side effects on the sequence of events. This chapter presents work toward a partial ordering and leaves logical constraints, such as preconditions and causation, as future work. Chapter 5 focuses on how temporal learning of this partial ordering can be accomplished.

2.3 Learning Narrative Relations

My model learns basic information about a narrative chain: the protagonist and the constituent subevents, although not their ordering. For this I need a metric for the relation between an event and a narrative chain.

Pairwise relations between events are first extracted unsupervised. A distributional score based on how often two events share grammatical arguments (using pointwise mutual information) is used to create this pairwise relation. Finally, a global narrative score is built such that all events in the chain provide feedback on the event in question (whether for inclusion or for decisions of inference).

Given a list of observed verb/dependency counts, I approximate the pointwise mutual information (PMI) by:

\[
\text{pmi}(\langle w, d \rangle, \langle v, g \rangle)) = \log \frac{P(\langle w, d \rangle , \langle v, g \rangle)}{P(\langle w, d \rangle )P(\langle v, g \rangle)}
\]  

(2.1)

where \( \langle w, d \rangle \) is the verb/dependency pair \( w \) and \( d \) (e.g., \( \langle \text{push}, \text{subject} \rangle \)). The numerator is defined by:

\[
P(\langle w, d \rangle , \langle v, g \rangle) = \frac{C(\langle w, d \rangle , \langle v, g \rangle)}{\sum_{x,y} \sum_{d,f} C(\langle x, d \rangle , \langle y, f \rangle)}
\]  

(2.2)

where \( C(\langle x, d \rangle , \langle y, f \rangle) \) is the number of times the two events \( \langle x, d \rangle \) and \( \langle y, f \rangle \) had a coreferring entity filling the values of the dependencies \( d \) and \( f \). I also adopt the
‘discount score’ from Pantel and Ravichandran (2004) to penalize low occurring words.

\[
pmi_d((w, d), (v, g)) = \frac{C((x, d), (y, f))}{C((x, d), (y, f)) + 1} \cdot \frac{\min(C((x, d), C((y, f))) + 1)}{\min(C((x, d), C((y, f))) + 1)}
\] (2.3)

Given the debate over appropriate metrics for distributional learning, I also experimented with the t-test. My experiments found that PMI outperforms the t-test on this task by itself and when interpolated together using various mixture weights.

Once pairwise relation scores are calculated, a global narrative score can then be built such that all events provide feedback on the event in question. For instance, given all narrative events from a chain in a document, I can find the next most likely event to occur by maximizing the chain’s score with each possible event.

\[
\text{chainscore}(C, (w, d)) = \sum_{(v, g) \in C} \text{pmi}((w, d), (v, g))
\] (2.4)

\[
\max_{j:0<j<m} \text{chainscore}(C, e_j)
\] (2.5)

where \(C\) is the chain of events and \(e_j\) is the jth event in the training corpus of \(m\) observed events. A ranked list of guesses can be built from this max and I hypothesize that the more events in the chain, the more informed the ranked output. An example of a chain with three events and the top six ranked guesses is given in figure 2.3.

### 2.3.1 Evaluation Metric: Narrative Cloze

The cloze task (Taylor, 1953) is used to evaluate a system (or human) for language proficiency by removing a random word from a sentence and having the system attempt to fill in the blank (e.g., I forgot to ___ the waitress for the good service). Depending on the type of word removed, the test can evaluate syntactic knowledge as well as semantic. Deyes (1984) proposed an extended task, discourse cloze, to
 CHAPTER 2. LEARNING NARRATIVE EVENT CHAINS

<table>
<thead>
<tr>
<th>Known events:</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨pleaded, subject⟩, ⟨admits, subject⟩, ⟨convicted, object⟩</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Likely Events:</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨sentenced, object⟩ 0.89</td>
</tr>
<tr>
<td>⟨paroled, object⟩ 0.76</td>
</tr>
<tr>
<td>⟨fired, object⟩ 0.75</td>
</tr>
</tbody>
</table>

Figure 2.3: Three narrative events and the six most likely events to include in the same chain.

evaluate discourse knowledge (removing phrases that are recoverable from knowledge of discourse relations like contrast and consequence).

I present a new cloze task that requires narrative knowledge to solve, the narrative cloze. The narrative cloze is a sequence of narrative events in a document from which one event has been removed. The task is to predict the missing verb and typed dependency. Take this example text about American football with McCann as the protagonist:

1. McCann threw two interceptions early.
2. Toledo pulled McCann aside and told him he’d start.
3. McCann quickly completed his first two passes.

These clauses are represented in the narrative model as the following five events:

⟨threw, subject⟩, ⟨pulled, object⟩, ⟨told, object⟩, ⟨start, subject⟩,
⟨completed, subject⟩.

These verb/dependency events make up a narrative cloze model. The model allows us to remove ⟨threw, subject⟩ and use the remaining four events to rank this missing event. Removing a single such pair to be filled in automatically allows us to evaluate a system’s knowledge of narrative relations and coherence. I do not claim this cloze task to be solvable even by humans, but rather assert it as a comparative measure to
evaluate narrative knowledge. Further, this task is particularly attractive for narrative chains because it aligns with one of the original ideas behind Schankian scripts, namely that scripts help humans ‘fill in the blanks’ when language is underspecified.

### 2.3.2 Narrative Cloze Experiment

I use years 1994-2004 (1,007,227 documents) of the New York Times section of the Gigaword Corpus, Third Edition (Graff, 2002) for training. The document count does not include duplicate news stories. I found up to 18% of the corpus are duplications, mostly AP reprints. I automatically found these by matching the first two paragraphs of each document, removing exact matches. I parse the text into typed dependency graphs with the Stanford Parser \(^1\) (de Marneffe et al., 2006), recording all verbs with subject, object, or prepositional typed dependencies. I use the OpenNLP\(^2\) coreference engine to resolve the entity mentions. For each document, the verb pairs that share coreferring entities are recorded with their dependency types. Particles are included with the verb.

I used 10 news stories from the 1994 section of the corpus for development. The stories were hand chosen to represent a range of topics such as business, sports, politics, and obituaries. I used 69 news stories from the 2001 (year selected randomly) section of the corpus for testing (also removed from training). The test set documents were randomly chosen and not preselected for a range of topics. From each document, the entity involved in the most events was selected as the protagonist. For this evaluation, I only look at verbs. All verb clauses involving the protagonist are manually extracted and translated into the narrative events \((verb, dependency)\). Exceptions that are not included are verbs in headlines, quotations (typically not part of a narrative), “be” properties (e.g., \textit{john is happy}), modifying verbs (e.g., \textit{hurried to leave}, only \textit{leave} is used), and multiple instances of one event.

The original test set included 100 documents, but those without a narrative chain at least five events in length were removed, leaving 69 documents. Most of the removed documents were not stories, but genres such as interviews and cooking recipes. An

\(^1\)\url{http://nlp.stanford.edu/software/lex-parser.shtml}
\(^2\)\url{http://opennlp.sourceforge.net}
New York Times Editorial

\( \langle \text{occupied, subject} \rangle \) \( \langle \text{brought, subject} \rangle \) \( \langle \text{rejecting, subject} \rangle \)

\( \langle \text{projects, subject} \rangle \) \( \langle \text{met, subject} \rangle \) \( \langle \text{appeared, subject} \rangle \)

\( \langle \text{offered, subject} \rangle \) \( \langle \text{voted, pp_for} \rangle \) \( \langle \text{offer, subject} \rangle \)

Figure 2.4: One of the 69 test documents, containing 10 narrative events. The protagonist is President Bush.

example of an extracted chain is shown in figure 2.4.

I evaluate with Narrative Cloze using leave-one-out cross validation, removing one event and using the rest to generate a ranked list of guesses. The test dataset produces 740 cloze tests (69 narratives with 740 events). After the model generates its ranked guesses, the position of the correct event is averaged over all 740 tests for the final score. I penalize unseen events by setting their ranked position to the length of the guess list (ranging from 2k to 15k). This harshly penalizes the missed guess by considering its rank as the end of the guess list.

Figure 2.3 is an example of a ranked guess list for a short chain of three events. If the original document contained \( \langle \text{fired, object} \rangle \), this cloze test would score 3.

Baseline

I want to measure the utility of the protagonist and the narrative coherence assumption, so the baseline learns relatedness strictly based upon verb co-occurrence. The PMI is then defined as between all occurrences of two verbs in the same document. This baseline evaluation is verb only, as dependencies require a protagonist to fill them.

After initial evaluations, the baseline was performing very poorly due to the huge amount of data involved in counting all possible verb pairs (using a protagonist vastly reduces the number). I experimented with various count cutoffs to remove rare occurring pairs of verbs. The final results use a baseline where all pairs occurring less
than 10 times in the training data are removed.

Since the verb-only baseline does not use typed dependencies, my narrative model cannot directly compare to this abstracted approach. I thus modified the narrative model to ignore typed dependencies, but still only count event pairs with shared arguments. Thus, I calculate the PMI across verbs that share arguments. This approach is called the Protagonist approach in the Results. The full narrative model with the protagonist that includes the grammatical dependencies is called Typed Deps.

Results
Experiments with varying sizes of training data are presented in figure 2.5. Each ranked list of candidate verbs for the missing event in the Baseline and Protagonist approaches contained approximately nine thousand candidates. Of the 740 cloze tests, 714 of the removed events were present in their respective list of guesses. This is encouraging as only 3.5% of the events are unseen (or do not meet cutoff thresholds).

When all training data is used (1994-2004), the average ranked position is 1826 for Baseline and 1160 for Protagonist (1 being most confident). The Baseline performs better at first (years 1994-5), but as more data is seen, the Baseline worsens while the Protagonist improves. This verb-only protagonist model shows a 36.5% improvement over the baseline trained on all years. Results from the full Typed Deps model, not comparable to the baseline, parallel the Protagonist results, improving as more data is seen (the average ranked position is 1908 with all of the training data; the Typed Deps line has lower overall scores because the number of possible ⟨verb, dependency⟩ events is much higher than verb-only). I also ran the experiment without OpenNLP coreference, and instead used exact and substring matching for coreference resolution. This showed a 5.7% absolute decrease in the verb-only results. These results show that a protagonist greatly assists in narrative judgements.

2.4 Discrete Narrative Event Chains
Up to this point, I have learned narrative relations across all possible events and their arguments. However, the discrete partitioned lists of events for which Schank scripts
Figure 2.5: Results with varying sizes of training data. Year 2003 is not explicitly shown because it has an unusually small number of documents compared to other years.
are most famous have not yet been constructed.

I intentionally did not set out to reproduce explicit self-contained *scripts* in the sense that the Restaurant Script is complete and cannot include other events. The name *narrative* was chosen to imply a *likely set* of events that is common in spoken and written retelling of world events. Discrete sets have the drawback of shutting out unseen and unlikely events from consideration. It is advantageous to consider a space of possible narrative events and the ordering within, not a closed list.

However, it is worthwhile to construct discrete narrative chains, if only to see whether the combination of event learning and ordering produce script-like structures. This is easily achievable by using the PMI scores from section 2.3 in an agglomerative clustering algorithm, and then applying the ordering algorithm that is described in chapter 5 to produce a directed graph.

Figures 2.6 and 2.7 show two learned chains after clustering and ordering. Each arrow indicates a before relation. Duplicate arrows implied by rules of transitivity are removed. Figure 2.6 is remarkably accurate, and figure 2.7 addresses one of the chains from this chapter’s introduction, the employment narrative. The core employment events are accurate, but clustering included life events (born, died, graduated) from obituaries of which some temporal information is incorrect. As Chapter 5 will address, my supervised temporal corpus does not include obituaries, thus I suffer from sparsity in training data.

### 2.5 Discussion

I have shown that it is possible to learn unordered narrative event chains without human intervention from raw text. Not only do the learned narrative relations show improvements over a baseline, but narrative chains offer hope for many other areas of NLP. Inference, coherence in summarization and generation, slot filling for question answering, and frame induction are all potential areas.

The main contribution of this chapter is the use of the protagonist to guide the learning process. The protagonist acts as a hook to extract a list of related events from each document, effectively learning a new measure of event similarity, the narrative
Figure 2.6: An automatically learned Prosecution Chain. Arrows indicate the before relation.
Figure 2.7: An Employment Chain. Dotted lines indicate incorrect \textit{before} relations.
relation. The 37% improvement over a verb-only baseline shows that pure distributional approaches based on bags-of-words are not as effective. The protagonist is a discourse-level relation that has not been explored in previous word similarity work and has implications for a range of NLP applications that rely on similarity judgements.

Second, the narrative chain representation is a significant contribution to knowledge representations for NLP. I showed how the event space of narrative relations can be clustered to create discrete sets of narrative event chains. While it is unclear if these are better than an unconstrained distribution of events, they do offer insight into the quality of narratives. This is the first work to learn both entities and events in a script-like representation. Large-scale learning of these chains show promise to inform more advanced NLP applications that require deeper reasoning. Chapter 6 describes in detail one such application: information extraction.

Finally, an important area not yet discussed is the possibility of using narrative chains for semantic role learning. A narrative chain can be viewed as defining the semantic roles of a chain of events, constraining it against roles of the other events in the chain. An argument’s class can then be defined as the set of narrative arguments in which it appears. I will define and expand on this idea in the next chapter.

This initial narrative chain model provides an important first step toward learning the rich causal, temporal and inferential structure of narrative schemas, and more broadly, of scripts and frames. The next chapter shows how to learn such a model: the narrative schema.
Chapter 3

Learning Narrative Schemas

This chapter extends the \textit{narrative event chain} representation to include all participants in its set of events, rather than a single participant. Narrative event chains from the previous chapter relied on the intuition that in a coherent text, any two events that are about the same participants are likely to be part of the same story or narrative. The model learned simple aspects of narrative structure (narrative chains) by extracting events that share a single participant, the \textit{protagonist}. This chapter now extends this approach to represent sets of situation-specific events and \textit{multiple} participants not unlike scripts, caseframes (Bean and Riloff, 2004), and FrameNet frames (Baker et al., 1998).

The core representation of this chapter and that which is used in the rest of the dissertation is the \textit{narrative schema}: coherent sequences or sets of events (\texttt{arrested(POLICE,SUSPECT)}, \texttt{convicted(JUDGE,SUSPECT)}) whose arguments are filled with participant semantic roles defined over words (\texttt{JUDGE} = \{judge, jury, court\}, \texttt{POLICE} = \{police, agent, authorities\}). This chapter will describe an algorithm to merge verbs in distinct narrative chains into an improved single narrative schema, while the shared arguments across verbs provide rich information for inducing semantic roles. As with narrative event chains, my approach does not use supervised techniques, hand-built knowledge, or predefined classes of events or roles. The unsupervised learning algorithm still observes coreferring arguments in chains of verbs, but learning will now take into account all such coreferring entities, rather than a
single protagonist, to now learn both rich narrative event structure and argument roles. By jointly addressing both tasks, I improve on the previous chapter’s results and induce richer frame-specific semantic roles for all entities and events.

The narrative event chain representation has two major limitations that this chapter addresses. First, the model only represents one participant (the protagonist). Representing the other entities involved in all event slots in the narrative could potentially provide valuable information. Second, the model does not express any information about the protagonist, such as its type or role. Role information (such as knowing whether a filler is a location, a person, a particular class of people, or even an inanimate object) could crucially inform learning and inference. I discuss both of these contributions here.

The Case for Joint Chains

The second problem with narrative chains is that they make judgments only between protagonist arguments, one slot per event. All entities and slots in the space of events should be jointly considered when making event relatedness decisions.

As an illustration, consider the verb arrest. Which verb is more related, convict or capture? A narrative chain might only look at the objects of these verbs and choose the one with the highest score, usually choosing convict. But in this case the subjects offer additional information; the subject of arrest (police) is different from that of convict (judge). A more informed decision prefers capture because both the objects (suspect) and subjects (police) are identical. This joint reasoning is absent from the narrative chain model.

The Case for Argument Types

Narrative event chains do not specify what type of argument fills the role of protagonist. Chain learning and clustering is based only on the frequency with which two verbs share arguments, ignoring any features of the arguments themselves. Let figure 3.1 serve as an example of an actual chain from an article in my training data. Given this chain of five events, I want to choose other events most likely to occur in
Figure 3.1: A narrative event chain and two as of yet unconnected events. Circles to the left of the verbs represent the syntactic subjects of the verbs, and the right represents objects. Edges indicate constraints between the syntactic functions that the entity must fill.

One of the top scoring event slots is (fly X). Narrative chains incorrectly favor (fly X) because it is observed during training with all five event slots, although not frequently with any one of them. An event slot like (charge X) is much more plausible, but is unfortunately scored lower by the model.

Representing the types of the arguments can help solve this problem. Few types of arguments are shared between the chain and (fly X). However, (charge X) shares many arguments with (accuse X), (search X) and (suspect X) (e.g., criminal and suspect). Even more telling is that these arguments are jointly shared (the same or coreferent) across all three events. Chains represent coherent scenarios, not just a set of independent pairs, so I want to model argument overlap across all pairs.

The Case for Semantic Roles

The task of semantic role learning and labeling is to identify classes of entities that fill predicate slots; semantic roles seem like they’d be a good model for the kind of argument types we’d like to learn for narratives. Most work on semantic role labeling, however, is supervised, using Propbank (Palmer et al., 2005), FrameNet (Baker et al., 1998) or VerbNet (Kipper et al., 2000) as gold standard roles and training data. More recent learning work has applied bootstrapping approaches (Swier and Stevenson,
2004a; He and Gildea, 2006a), but these still rely on a hand labeled seed corpus as well as a pre-defined set of roles. Grenager and Manning (2006a) use the EM algorithm to learn PropBank roles from unlabeled data, and unlike bootstrapping, they don’t need a labeled corpus from which to start. However, they do require a predefined set of roles (arg0, arg1, etc.) to define the domain of their probabilistic model.

Green and Dorr (2005) use WordNet’s graph structure to cluster its verbs into FrameNet frames, using glosses to name potential slots. I differ in that I attempt to learn frame-like narrative structure from untagged newspaper text. Most similar to us, Alishahi and Stevenson (2007) learn verb specific semantic profiles of arguments using WordNet classes to define the roles. I learn situation-specific classes of roles shared by multiple verbs.

Thus, two open goals in role learning include (1) unsupervised learning and (2) learning the roles themselves rather than relying on pre-defined role classes. As just described, narrative chains offer an unsupervised approach to event learning (goal 1), but lack semantic role knowledge (goal 2). The following sections describe a model that addresses both goals.

### 3.1 Narrative Schemas

The next sections introduce typed narrative chains and chain merging, extensions that allow us to jointly learn argument roles with event structure.

#### 3.1.1 Typed Narrative Chains

The first step in describing a narrative schema is to extend the definition of a narrative chain to include argument types. I now constrain the protagonist to be of a certain type or role. A Typed Narrative Chain is a partially ordered set of event slots that share an argument, but now the shared argument is a role defined by being a member of a set of types $R$. These types can be lexical units (such as observed head words), noun clusters, or other semantic representations. I use head words in
the examples below, but I also evaluate with argument clustering by mapping head words to member clusters created with the CBC clustering algorithm (Pantel and Lin, 2002).

I define a typed narrative chain as a tuple $(L, P, O)$ with $L$ and $O$ the set of event slots and partial ordering as before. Let $P$ be a set of argument types (head words) representing a single role. An example is given here:

$$L = \{(\text{hunt } X), (\text{X use}), (\text{suspect } X), (\text{accuse } X), (\text{search } X)\}$$

$$P = \{\text{person}, \text{government}, \text{company}, \text{criminal}, \ldots\}$$

$$O = \{(\text{use, hunt}), (\text{suspect, search}), (\text{suspect, accuse}) \ldots\}$$

### 3.1.2 Learning Argument Types

As mentioned above, narrative chains are learned by parsing the text, resolving coreference, and extracting chains of events that share participants. In my new model, argument types are learned simultaneously with narrative chains by finding salient words that represent coreferential arguments. I record counts of arguments that are observed with each pair of event slots, build the referential set for each word from its coreference chain, and then represent each observed argument by the most frequent head word in its referential set (ignoring pronouns and mapping entity mentions with person pronouns to a constant PERSON identifier). As an example, the following contains four *worker* mentions:

But for a growing proportion of **U.S. workers**, the troubles really set in when **they** apply for unemployment benefits. Many **workers** find **their** benefits challenged.

The four bolded terms are coreferential and (hopefully) identified by coreference. My algorithm chooses the head word of each phrase and ignores the pronouns. It then chooses the most frequent head word as the most salient mention. In this example, the most salient term is *workers*. If any pair of event slots share arguments from this set, I count *workers*. In this example, the pair (X find) and (X apply) shares an argument (they and workers). The pair ((X find),(X apply)) is counted once for narrative chain induction, and ((X find), (X apply), workers) once for argument induction.
Figure 3.2: A typed narrative chain. The four top arguments are given. The ordering $O$ is not shown.

Figure 3.2 shows the top occurring words across all event slot pairs in a criminal scenario chain. This chain will be part of a larger narrative schema, described in section 3.1.4.

### 3.1.3 Event Slot Similarity with Arguments

I now formalize event slot similarity with arguments. Narrative chains as defined above in Chapter 2 score a new event slot $\langle f, g \rangle$ against a chain of size $n$ by summing over the scores between all pairs:

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

(3.1)

where $C$ is a narrative chain, $f$ is a verb with grammatical argument $g$, and $\text{sim}(e, e')$ is the pointwise mutual information $\text{pmi}(e, e')$. Growing a chain by one adds the highest scoring event.

I extend this function to include argument types by defining similarity in the context of a specific argument $a$:

$$\text{sim}(\langle e, d \rangle, \langle e', d' \rangle, a) =$$

$$\text{pmi}(\langle e, d \rangle, \langle e', d' \rangle) + \lambda \log \text{freq}(\langle e, d \rangle, \langle e', d' \rangle, a)$$

(3.2)

where $\lambda$ is a constant weighting factor and $\text{freq}(b, b', a)$ is the corpus count of $a$ filling the arguments of events $b$ and $b'$. I performed a grid search to set $\lambda = .08$ to maximize the results on the development set in the narrative cloze evaluation, discussed later.
I then score the entire chain for a particular argument:

\[
\text{score}(C, a) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{sim}(\langle e_i, d_i \rangle, \langle e_j, d_j \rangle, a) \quad (3.3)
\]

Using this chain score, I finally extend \textit{chainsim} to score a new event slot based on the argument that maximizes the entire chain’s score:

\[
\text{chainsim}'(C, \langle f, g \rangle) = \max_a (\text{score}(C, a) + \sum_{i=1}^{n} \text{sim}(\langle e_i, d_i \rangle, \langle f, g \rangle, a)) \quad (3.4)
\]

The argument is now directly influencing event slot similarity scores. I will use this definition in the next section to build Narrative Schemas.

### 3.1.4 Narrative Schema: Multiple Chains

Whereas a narrative chain is a set of event slots, a Narrative Schema is a set of typed narrative chains. A schema thus models all actors in a set of events. If \textit{(push X)} is in one chain, \textit{(Y push)} is in another. This allows us to model a document’s entire narrative, not just one main actor.

#### The Model

A narrative schema is defined as a 2-tuple \( N = (E, C) \) with \( E \) a set of events (here defined as verbs) and \( C \) a set of typed chains over the event slots. I represent an event as a verb \( v \) and its grammatical argument positions \( D_v \subseteq \{\text{subject, object, prep}\} \). Thus, each event slot \( \langle v, d \rangle \) for all \( d \in D_v \) belongs to a chain \( c \in C \) in the schema. Further, each \( c \) must be unique for each slot of a single verb. Using the criminal prosecution domain as an example, a narrative schema in this domain is built as in figure 3.3.

The three dotted boxes are graphical representations of the typed chains that are combined in this schema. The first represents the event slots in which the criminal
CHAPTER 3. LEARNING NARRATIVE SCHEMAS

is involved, the second the police, and the third is a court or judge. Although my representation uses a set of chains, it is equivalent to represent a schema as a constraint satisfaction problem between \( \langle e, d \rangle \) event slots. The next section describes how to learn these schemas.

Learning Narrative Schemas

Previous work on narrative chains focused on relatedness scores between pairs of verb arguments (event slots). The clustering step which built chains depended on these pairwise scores. Narrative schemas use a generalization of the entire verb with all of its arguments. A joint decision can be made such that a verb is added to a schema if both its subject and object are assigned to chains in the schema with high confidence.

For instance, it may be the case that \( (Y \text{ pull over}) \) scores well with the ‘police’ chain in figure 3.4. However, the object of \( (\text{pull over } A) \) is not present in any of the other chains. Police pull over cars, but this schema does not have a chain involving cars. In contrast, \( (Y \text{ search}) \) scores well with the ‘police’ chain and \( (\text{search } X) \) scores well in the ‘defendant’ chain too. Thus, I want to favor search instead of pull over because the schema is already modeling both arguments.

This intuition leads us to my event relatedness function for the entire narrative schema \( N \), not just one chain. Instead of asking which event slot \( \langle v, d \rangle \) is a best fit, I ask if \( v \) is best by considering all slots at once:
\[ narsim(N, v) = \sum_{d \in D_v} \max(\beta, \max_{c \in C_N} \text{chainsim}'(c, \langle v, d \rangle)) \] (3.5)

where \( C_N \) is the set of chains in the narrative \( N \). If \( \langle v, d \rangle \) does not have strong enough similarity with any chain, it creates a new one with base score \( \beta \). The \( \beta \) parameter balances this decision of adding to an existing chain in \( N \) or creating a new one. Based on experiments on the development set of the narrative cloze evaluation (upcoming in Section 3.5), a \( \beta \) value that seemed to discourage creating new chains performed the best. The value used for this section’s experiments is \( \beta = 0.2 \).

**Building Schemas**

I use equation 3.5 to build schemas from the set of *events* as opposed to the set of *event slots* that I previously used to learn individual narrative chains. In Chapter 2, narrative chains added the best verb-dependency pair \( \langle v_j, g_j \rangle \) based on the following:

\[ \max_{j: 0 < j < m} \text{chainsim}(c, \langle v_j, g_j \rangle) \] (3.6)

where \( m \) is the number of seen event slots in the corpus and \( \langle v_j, g_j \rangle \) is the jth such possible event slot. Schemas are now learned by adding events that maximize equation 3.5:

\[ \max_{j: 0 < j < |v|} narsim(N, v_j) \] (3.7)

where \( |v| \) is the number of observed verbs and \( v_j \) is the jth such verb. Verbs are incrementally added to a narrative schema by strength of similarity.

**3.2 Sample Narrative Schemas**

Figures 3.4 and 3.5 show two criminal schemas learned completely automatically from the NYT portion of the Gigaword Corpus, Third Edition (Graff, 2002). I parse the text into dependency graphs and resolve coreferences. The figures result from learning over the event slot counts. In addition, table 3.1 shows five of the top fifty
Figure 3.4: Graphical view of an unordered schema automatically built starting from the verb ‘arrest’. A $\beta$ value that encouraged splitting was used for this example. I ultimately set $\beta = 0.2$ by optimizing cloze performance on the development dataset.

scoring narrative schemas learned by my system. I artificially required the clustering procedure to stop (and sometimes continue) at six events per schema. Six was chosen as the size to enable us to compare to FrameNet in the next section; the mean number of verbs in FrameNet frames is between five and six. I use the optimized $\beta = 0.2$ as discussed above. I built a new schema starting from each verb that occurs in more than 3000 and less than 50,000 documents in the NYT section. This amounted to approximately 1800 verbs from which I show six varying domains captured in the top 20. Not surprisingly, most of these top schemas concern business, politics, crime, or food.

To further illustrate the range of domains that this learning process captures, Figures 3.6 and 3.7 contain hand-selected schemas across a variety of domains. These are some of the hundreds of schemas available online in my database of learned schemas.\footnote{http://cs.stanford.edu/people/nc/data/schemas/acl09/}
Table 3.1: Five of the top fifty scored Narrative Schemas. Events and arguments in italics were marked misaligned by FrameNet definitions. * indicates verbs not in FrameNet. - indicates verb senses not in FameNet.
CHAPTER 3. LEARNING NARRATIVE SCHEMAS

3.3 Evaluation: Frames and Roles

Most previous work on unsupervised semantic role labeling assumes that the set of possible classes is very small (i.e., PropBank roles \texttt{arg0} and \texttt{arg1}) and is known in advance. By contrast, my approach induces sets of entities that appear in the argument positions of verbs in a narrative schema. My model thus does not assume the set of roles is known in advance, and it learns the roles at the same time as clustering verbs into frame-like schemas. The resulting sets of entities (such as \{\texttt{police}, \texttt{agent}, \texttt{authorities}, \texttt{government}\} or \{\texttt{court}, \texttt{judge}, \texttt{justice}\}) can be viewed as a kind of schema-specific semantic role.

How can this unsupervised method of learning roles be evaluated? In Section 3.5 I evaluate the schemas together with their arguments in a cloze task. In this section I perform a more qualitative evaluation by comparing my schemas to FrameNet.

FrameNet (Baker et al., 1998) is a database of frames, structures that characterize particular situations. A frame consists of a set of events (the verbs and nouns that describe them) and a set of frame-specific semantic roles called \textit{frame elements} that can be arguments of the lexical units in the frame. FrameNet frames share commonalities with narrative schemas; both represent aspects of situations in the world, and both link semantically related words into frame-like sets in which each predicate
CHAPTER 3. LEARNING NARRATIVE SCHEMAS

Medical (Viral)
Events: infect transmit cause spread contract carry kill detect

Role 1: \{ transmit-o kill-s contract-o carry-o cause-s detect-o spread-s \}
\{ virus disease bacteria cancer toxoplasma strain fire parasite \}

Role 2: \{ detect-s kill-o spread-o carry-s transmit-o infect-o contract-s cause-o \}
\{ mosquito aids virus tick catastrophe disease aboard mite others \}

Financial
Verbs: cut raise reduce lower increase boost trim slash

Role 1: \{ raise-s cut-s increase-s reduce-s slash-s trim-s boost-s lower-s \}
\{ company fed bank government rates bundesbank plan bill \}

Role 2: \{ slash-o trim-o boost-o lower-o raise-o reduce-o cut-o increase-o \}
\{ rates rate price tax risk dividend stake estimate rating \}

Legal
Events: prohibit violate require allow bar forbid ban permit

Role 1: \{ violate-o forbid-s ban-s bar-s require-s allow-s prohibit-s permit-s \}
\{ law bill rule amendment act treaty constitution laws policy \}

Role 2: \{ ban-o bar-o require-o permit-o forbid-o allow-o violate-s prohibit-o \}
\{ company microsoft government iraq state use group banks student \}

Criminal
Events: arrest raid search detain found charge seize identify

Role 1: \{ detain-s found-s seize-s raid-s search-s charge-s identify-s arrest-s \}
\{ police agent authorities officer official investigator fbi troops soldier \}

Role 2: \{ identify-o charge-o arrest-o raid-o seize-o detain-o found-o search-o \}
\{ suspect police padilla officer yates driver government member citizen \}

Figure 3.6: Narrative schemas: examples were hand selected from the database to illustrate the diversity of learned narratives.
CHAPTER 3. LEARNING NARRATIVE SCHEMAS

Authorship
Events: publish sell write translate distribute edit produce read

Role 1: \{ translate-s produce-s sell-s write-s distribute-s publish-s read-s edit-s \}
\{ company author group year microsoft magazine my time firm writer \}

Role 2: \{ produce-o edit-o sell-o translate-o publish-o read-o write-o distribute-o \}
\{ book report novel article story letter magazine film letters movie show \}

Sports
Events: outscore outshot outrebounded beat score outplay trail tie

Role 1: \{ beat-s tie-s outplay-s score-s outrebounded-s outscore-s outshot-s trail-s \}
\{ king maverick sonics ranger lakers bruin angel dodger mets yankee \}

Role 2: \{ beat-o tie-o score-o outrebounded-o outscore-o outshot-o outplay-o... \}
\{ knicks king net maverick lakers state point patriot yankee jet celtic \}

Legislative
Events: veto pass oppose approve sign support require sponsor

Role 1: \{ sign-s oppose-s approve-s require-o veto-s sponsor-s support-s pass-s \}
\{ clinton bill house bush president state congress voter governor group \}

Role 2: \{ sponsor-o require-s veto-o pass-o approve-o oppose-o support-o sign-o \}
\{ bill legislation measure law amendment plan treaty agreement... \}

Culinary
Events: peel slice chop cook saute boil cut add

Role 1: \{ cut-s boil-s add-s slice-s peel-s saute-s cook-s chop-s \}
\{ wash thinly heat plan company potato cool finn remove measure \}

Role 2: \{ peel-o slice-o boil-o add-o cook-o chop-o cut-o saute-o \}
\{ potato onion tomato beet cucumber mushroom shrimp sample eggs steak \}

Figure 3.7: Narrative schemas: examples were hand selected from the database to illustrate the diversity of learned narratives.
draws its argument roles from a frame-specific set. They differ in that schemas focus on events in a narrative, while frames focus on events that share core participants. Nonetheless, the fact that FrameNet defines frame-specific argument roles suggests that comparing my schemas and roles to FrameNet would be elucidating.

I evaluate the top 50 learned narrative schemas, ordered as described in the previous section, and use FrameNet to perform qualitative evaluations on three aspects of schema: verb groupings, linking structure (the mapping of each argument role to syntactic subject or object), and the roles themselves (the set of entities that constitutes the schema roles).

**Verb groupings** To compare a schema’s event selection to a frame’s lexical units, I first map the top 50 schemas to the FrameNet frames that have the largest overlap with each schema’s six verbs. I was able to map 34 of the 50 narratives to FrameNet (for the remaining 16, no frame contained more than one of the six verbs). The remaining 34 schemas contained 6 verbs each for a total of 204 verbs. 40 of these verbs, however, did not occur in FrameNet, either at all, or with the correct sense. Of the remaining 164 verb mappings, 103 (63%) occurred in the closest FrameNet frame or in a frame one link away. 61 verbs (37%) thus occurred in a different frame than the one chosen.

I examined the 37% of verbs that occurred in a different frame. Most occurred in related frames, but did not have FrameNet links between them. For instance, one schema includes the causal verb *trade* with unaccusative verbs of change like *rise* and *fall*. FrameNet separates these classes of verbs into distinct frames, distinguishing motion frames from caused-motion frames. Even though *trade* and *rise* are in different FrameNet frames, they do in fact have the narrative relation that my system discovered. Of the 61 misaligned events, I judged all but one or two to be plausible in a narrative sense. Thus although not exactly aligned with FrameNet’s notion of event clusters, my induction algorithm seems to do very well.

**Linking structure** Next, I compare a schema’s linking structure, the grammatical relation chosen for each verb event. I thus decide, e.g., if the object of the verb arrest
(arrest B) plays the same role as the object of detain (detain B), or if the subject of detain (B detain) would have been more appropriate.

I evaluated the clustering decisions of the 34 schemas (204 verbs) that mapped to frames. For each chain in a schema, I identified the frame element that could correctly fill the most verb arguments in the chain. The remaining arguments were considered incorrect. Because I assumed all verbs to be transitive, there were 408 possible arguments (subjects and objects) in the 34 schemas. Of these 408 arguments, 386 were correctly clustered together, achieving 94.6% accuracy.

The schema in table 3.1 with events detain, seize, arrest, etc. shows some of these errors. The object of all of these verbs is an animate theme, but confiscate B and raid B are incorrect; people cannot be confiscated/raided. They should have been split into their own chain within the schema.

**Argument Roles** Finally, I evaluate the learned sets of entities that fill the argument slots. As with the above linking evaluation, I first identify the best frame element for each argument. For example, the events in the first schema of table 3.1 map to the Manufacturing frame. Argument B was identified as the Product frame element. I then evaluate the top 10 arguments in the argument set, judging whether each is a reasonable filler of the role. In my example, drug and product are correct Product arguments. An incorrect argument is test, as it was judged that a test is not a product.

I evaluated all 50 schemas. The 34 mapped schemas used their assigned frames, and I created frame element definitions for the remaining 16 that were consistent with the syntactic positions. There were 869 guessed arguments (50 schemas, top 20 max in 2 chains of each schema), and 659 were judged correct for a precision of 76%. This number includes Person and Organization names as correct fillers.

Most of the errors appear to be from parsing mistakes. Several resulted from confusing objects with adjuncts. Others misattached modifiers, such as including most as an argument. The cooking schema appears to have attached verbal arguments learned from instruction lists (wash, heat, boil). Two schemas require situations as arguments, but the dependency graphs chose as arguments the subjects of the
3.4 Evaluation: Coverage

The previous FrameNet evaluation showed agreement between our learned schemas and a human created database. However, a concern for any automatically acquired knowledgebase of schemas is its relevance to and coverage over new data. We want to measure the extent to which this dissertation’s narrative schema database can explain the events described in newspaper articles.

3.4.1 Event Coverage

This evaluation measures the amount of overlap between the learned schemas and the naturally occurring sets of events in documents. Newspaper articles describe sequences of events and often contain several narrative schemas. Chapter 2 defined a document’s central entity as the protagonist and manually labeled a set of documents for the main narrative chain involving only that actor. While I evaluated those narrative instances for an event prediction task, I now make use of the same data to measure how much of the chain is covered by the full narrative schemas. My learned database contains generalized schemas, and so I do not expect all chains to be covered as documents describe very specific narrative instances, however, the goal is to discover the extent of overlap with the database.

Data

I use the narrative chain test set as described in the cloze evaluation in chapter 2. The test set includes 69 randomly selected documents from the 2001 NYT portion of the Gigaword Corpus, Third Edition Graff (2002). The most repeated entity in each document is labeled as the protagonist, and all verbs of which he/she is the subject, object or preposition phrase are hand extracted to represent the narrative chain (note that this is not a full schema since it extracts a single entity and only
the grammatical positions that it fills). Verbs with low IDF scores\(^2\) are ignored. This data is also available online\(^3\).

**Overlap Metric**

A narrative chain is a set of connected events, so I want to measure the connectivity between these same events and my learned schemas. An event in a narrative chain is a predicate \(p\) (e.g., arrest) and a syntactic position \(d\) (e.g., subject or object). In a test document, there are edges between all events involving the protagonist, so the events are fully connected by definition.

In my learned schemas, an edge exists between two events if there exists some narrative schema such that one of its roles contains both events. I define coverage as a graph connectivity problem. Given a set of events from a document, how connected is the set in my database of learned schemas?

There are several options for measuring connectivity in graph theory. I adopt the largest connected component approach for this analysis. A connected component is a subset of vertices such that a path exists between every vertex in the subset. For each test document’s connected events (the narrative chain), I compute the largest connected component in our database’s edges and return the number of vertices in the component. For a test chain of length \(n\), returning \(n\) indicates full coverage by the database (there exists a schema such that all \(n\) events are members). Returning zero means that none of the events in the test chain appear together in a single schema.

As a concrete example, consider a newspaper article describing someone who has written and published a book, using verbs like *write*, *edit*, *publish*, *distribute*, and *sell*. Considering the protagonist of such an article as the book being written, the test set will include the narrative chain as follows: *write-obj*, *edit-obj*, *publish-obj*, *distribute-obj*, *sell-obj*. These object positions are the nodes of the graph, and we consider it fully connected. This evaluation finds the single narrative schema with the largest connected component including these verbs. If there is a schema with four of the five verbs connected by their objects, the score is \(4/5 = 80\%\).

\(^2\)0.9 threshold, removing any below it

\(^3\)http://cs.stanford.edu/people/nc/data/chains
3.4.2 Results

For each test document, the size of the largest connected component is computed and the percentage of events covered by that component is returned. For instance, if a document has 10 events in its narrative chain and the database contains a narrative schema that includes 7 of the events, 70% is our coverage over that document. We macro-average these percents across all 69 documents’ 740 events. The final percent coverage for my learned schema database using the database with schemas of size 12 is 34.0%.

The database of narrative schemas thus connects approximately one out of every three events in newspaper articles. In other words, one third of a document’s events are part of a single self-contained narrative schema. Since schemas characterize general sequences of events, it is expected that a significant portion of an article’s events would occur outside a single schema, or the story would not be newsworthy for conveying new information. This suggests that over one third of a news story is about common events, with the remaining two thirds containing new information in which the reader may be interested. Chapter 2 found that only 3.5% of the events were completely unconnected to other events in the space of seen event pairs. This further suggests that the two thirds of news events are connected, but not prevalent enough to draw generalizations in the form of a narrative schema and may be situation-specific. How many of these can still be learned through further advances remains for future work.

3.5 Evaluation: Cloze

The previous section compared my learned knowledge to current work in event and role semantics. I now provide a more formal evaluation against untyped narrative chains. The two main contributions of schemas are (1) adding typed arguments and (2) considering joint chains in one model. I evaluate each using the narrative cloze test as in Chapter 2.
3.5.1 Narrative Cloze

The cloze task (Taylor, 1953) evaluates human understanding of lexical units by removing a random word from a sentence and asking the subject to guess what is missing. The narrative cloze is a variation on this idea that removes an event slot from a known narrative chain. Performance is measured by the position of the missing event slot in a system’s ranked guess list. I presented this new evaluation framework in detail in Section 2.3.1. As discussed earlier, this task is particularly attractive for narrative schemas (and chains) because it aligns with one of the original ideas behind Schankian scripts, namely that scripts help humans ‘fill in the blanks’ when language is underspecified.

3.5.2 Training and Test Data

I count verb pairs and shared arguments over the NYT portion of the Gigaword Corpus, Third Edition (years 1994-2004), approximately one million articles. I parse the text into typed dependency graphs with the Stanford Parser (de Marneffe et al., 2006), recording all verbs with subject, object, or prepositional typed dependencies. Unlike in Section 2.3.1, I lemmatize verbs and argument head words. I use the OpenNLP coreference engine to resolve entity mentions.

The test set is the same as in (Chambers and Jurafsky, 2008b). 100 random news articles were selected from the 2001 NYT section of the Gigaword Corpus. Articles that did not contain a protagonist with five or more events were ignored, leaving a test set of 69 articles. I used a smaller development set of size 17 to tune parameters.

3.5.3 Typed Chains

The first evaluation compares untyped against typed narrative event chains. The typed model uses equation 3.4 for chain clustering. The dotted line ‘Chain’ and solid ‘Typed Chain’ in figure 3.8 shows the average ranked position over the test set. The untyped chains plateau and begin to worsen as the amount of training data increases,
but the typed model is able to improve for some time after. I see a 6.9\% gain at 2004 when both lines trend upwards.

### 3.5.4 Narrative Schema

The second evaluation compares the performance of the narrative schema model against single narrative chains. I ignore argument types and use untyped chains in both (using equation 1 instead of 4). The dotted line ‘Chain’ and solid ‘Schema’ show performance results in figure 3.8. Narrative Schemas have better ranked scores in all data sizes and follow the previous experiment in improving results as more data is added even though untyped chains trend upward. I see a 3.3\% gain at 2004.

### 3.5.5 Typed Narrative Schema

The final evaluation combines schemas with argument types to measure overall gain. I evaluated with both head words and CBC clusters as argument representations. Not only do typed chains and schemas outperform untyped chains, combining the two gives a further performance boost. Clustered arguments improve the results further, helping with sparse argument counts (‘Typed Schema’ in figure 3.8 uses CBC arguments). Overall, using all the data (by year 2004) shows a 10.1\% improvement over untyped narrative chains.

### 3.6 Discussion

The main contributions in this chapter are threefold: a joint model for all entities in the schema, learning typed arguments (rather than just constraints between an unknown protagonist), and a characterization of the semantic roles in a broader situation’s context. Further, this is the first fully automatic approach to learning such structured knowledge about events and entities from raw text.

The significant improvement in the cloze evaluation shows that even though narrative cloze does not evaluate argument types, jointly modeling the arguments with events improves event clustering. Likewise, the FrameNet evaluation shows that
Figure 3.8: Results on the narrative cloze test with varying sizes of training data. Both narrative chains and narrative schemas are compared with their untyped and typed versions.
learned schemas are precise as compared against human-created frames: over 94% accuracy in syntax to role links, and 76% accuracy in selectional preferences. Likewise, this FrameNet comparison suggests that modeling related events assists argument learning, and vice versa. The tasks mutually inform each other.

Further, my new argument learning algorithm carries implications for semantic role learning. Not only does it perform unsupervised induction of situation-specific role classes, but the resulting roles and linking structures may also offer the possibility of (unsupervised) FrameNet-style semantic role labeling. Although this learned representation is not the traditional verb-specific role labeling task, this latter finding is one of the first attempts toward an unsupervised induction of situation-specific role classes.

Finding the best argument representation is an important future direction. Narrative schemas use a basic representation of noun clusters, or bag of words, observed coreferring between the event pairs. The performance of these typed schemas in figure 3.8 showed that while the other non-typed approaches leveled off, the typed schemas continually improved with more data. The exact balance between lexical units, clusters, or more general (traditional) semantic roles remains to be solved, and may be application specific.

This chapter’s learning algorithm contains several parameters that control schema learning. The maximum size of the learned schemas is an important parameter, although it is quite flexible and may vary based on the end application’s needs. The $\beta$ parameter is perhaps the most important as it controls when an argument starts a new chain, or joins with an already established chain. I found that the cloze evaluation performed best when $\beta$ discouraged new chains. This is why most of the learned schemas have only two chains, combining transitive verbs with the same two arguments. A lower $\beta$ creates diverse schemas with more chains, but at the expense of the evaluations. Future work needs to study what types of situations are being missed as a result, and how this affects end-user applications such as Information Extraction (Chapter 6).

Finally, one property of narrative schemas not addressed in this chapter is the ordering of events within schemas. I have shown how to induce event structure and
argument roles, but leave ordering to be integrated separately via the supervised algorithm in Chapter 4. Finally, as discussed in Chapter 1, a range of natural language understanding applications can benefit from the rich inferential structures that narrative schemas provide. Chapter 6, in particular, will perform a complete information extraction task that applies learned narrative schemas to extract and understand the information contained in newspaper articles.
Chapter 4

Learning to Order Events

Up to this point, I have described how to learn unordered narrative schemas from raw text. This chapter now addresses the task of learning the event orderings within the schemas. Knowing the order of events can enable many important NLP applications. For example, knowing that convictions occur after arrests enables systems to reason over causation, infer future events, and recover a document’s original timeline. However, in order to learn general event orders, I first need to learn to order specific instances of events as they occur in documents. This chapter describes two approaches to building supervised classifiers that can make such within-document decisions. I will describe two significant contributions to the field of event ordering: a state-of-the-art pairwise classifier for events, and the first global constraint model over a document’s entire context.

The first significant contribution in this chapter is a fully automatic machine learning architecture that learns temporal relations between pairs of events. The first stage learns the temporal attributes of single event descriptions, such as tense, grammatical aspect, and aspectual class. These imperfect guesses, combined with other linguistic features, are then sent to a second stage that classifies the temporal relationship between two events. I analyze my new features and present results on the TimeBank Corpus that are 3% higher than previous work that used perfect human tagged features. This was the first system to classify the order of event pairs without gold features.
Second, this chapter describes the first global model that constrains a document’s possible event orderings. Most work on event-event ordering has focused on improving classifiers for pairwise decisions, ignoring obvious contradictions in the global space of events when misclassifications occur. My global framework to repair these event ordering mistakes was the first to be proposed for this task. This chapter addresses three main factors involved in a global framework: the global optimization algorithm, the constraints that are relevant to the task, and the level of connectedness across pairwise decisions. I employ Integer Linear Programming to address the first factor, drawing from related work in paragraph ordering (Bramsen et al., 2006). After finding minimal gain with the initial model, I explore reasons for and solutions to the remaining two factors through temporal reasoning and transitivity rule expansion.

This chapter thus accomplishes both aspects of learning event order: a local classifier for event-event pairs, and a global model that integrates the local classifier’s scores to maintain consistency and maximize the document’s overall predictions. I begin with a brief review of previous work, and then present the pairwise and global models in order.

4.1 Previous Work

The creation of the Timebank Corpus (Pustejovsky et al., 2003) facilitated the development of machine learning techniques for event ordering tasks, and provided a common evaluation with which to compare. The corpus labels verbs and nouns as events, and then annotates the time relations between event pairs (e.g., before, after, simultaneous, includes, etc.). Several research thrusts have since approached this event-event ordering task with supervised learning. Mani et al. (2007) built a Max-Ent classifier that assigns each pair of events one of six relations from an augmented Timebank corpus. Their classifier relied on gold features from the corpus annotations, including verb tense, aspect, modality, polarity and event class. Pairwise agreement on tense and aspect are also included (e.g., do both events occur in the past?). In a second study, they applied rules of temporal transitivity to greatly expand the corpus, providing better results on this enlarged dataset.
CHAPTER 4. LEARNING TO ORDER EVENTS

The TempEval challenges (Verhagen et al., 2007, 2009) have also spurred increased work in this area, focusing on learning the ordering between events, time expressions, and the document timestamp. Approaches to solving the ordering task range from rule-based systems (Hagege and Tannier, 2007; Puscasu, 2007), to feature-based supervised classifiers (Min et al., 2007; Hepple et al., 2007; Bethard and Martin, 2007), to an HMM model of the events as they occur in textual order (Cheng et al., 2007). The contest uses two relations, before and after, in a semi-complete textual classification task with a new third relation to distinguish relations that can be labeled with high confidence from those that are uncertain, called vague. The task was a simplified classification task from Timebank in that only one verb, the main verb, of each sentence was used. Thus, the task can be viewed as ordering the main events in pairwise sentences rather than the entire document.

Bethard et al. (2007) took a large subset of Timebank and explored the event space for event-event pairs occurring in verb-clause relationships. They hand-tagged every pair of events in these relationships, forcing the annotators to use an overlap label when no before/after relationship was clear. This differed from Timebank in that Timebank annotators were never required to label any specific pairs. Timebank also does not have a more general overlap relation. They built custom features and achieved 89% accuracy for this subtask. I will look at both this smaller set of three relations, and the full set of Timebank relations.

In contrast to work on the Timebank Corpus, Lapata and Lascarides (2006) trained an event classifier for inter-sentential events. They built a corpus by saving sentences that contained two events, one of which is triggered by a key time word (e.g., after and before). Their learner was based on syntax and clausal ordering features. Boguraev and Ando (2005) evaluated machine learning on related tasks, but not relevant to event-event classification.

My work is most similar to the supervised approaches on Timebank and TempEval in that I learn relations between event pairs, but my work extends their results both with new features and by using fully automatic linguistic features from raw text that are not hand selected as gold values from an annotated corpus.

As the above work on ordering events has focused on local pairwise decisions,
these approaches ignored globally inconsistent labels. However, temporal ordering is the type of domain in which global constraints should be relatively easy to represent and reason over. Bramsen et al. (2006) proposed such a model to assist in ordering paragraphs of text (not atomic events) in the medical domain. They found that Integer Linear Programming (ILP) outperformed other methods. I build on this work to predict globally consistent orderings. I explore how to use time expressions to connect events that are otherwise disconnected in the document, and augment the global graph of events to encourage the ILP algorithm to enforce consistency. Since publication of this global model (Chambers and Jurafsky, 2008a), others have developed complementary global models using markov logic based on my approach (Yoshikawa et al., 2009).

This chapter thus solves the original Timebank task of predicting event-event pair orders over all six relations (e.g., before, immediately before, includes, simultaneous, begins, ends), being the first to do this without gold features. I then use the three core relations of TempEval (before, after, vague) and build a new global architecture that applies them to the full document ordering task. I extend this previous work by describing the first temporal reasoning component and embedding it within a global constraint model using ILP.

4.2 Pairwise Classification

Initial work on the Timebank Corpus revealed that the six-class classification of temporal relations is very difficult, even for human annotators. The initial supervised classification scores reported on Timebank achieved 62.5% accuracy when using gold-standard features as marked by humans (Mani et al., 2006). I will describe an approach that uses features extracted automatically from the raw text that not only duplicates this performance, but surpasses its absolute accuracy by 3%. I do so through advanced linguistic features and a surprising finding that using automatic rather than hand-labeled tense and aspect knowledge causes only a slight performance degradation.

Section 4.2.2 describes the first stage of basic temporal extraction, followed by a
full description of the second stage in 4.2.3. The evaluation and results on Timebank then follow in section 4.2.4.

4.2.1 Data: The Timebank Corpus

I use the Timebank Corpus (v1.1) for evaluation. The Timebank Corpus (Pustejovsky et al., 2003) is a corpus of 186 newswire articles that are tagged for events, time expressions, and relations between the events and times. There are 3345 individual events that are further tagged for temporal information such as tense, modality and grammatical aspect. Time expressions use the TimeML (Ingria and Pustejovsky, 2002) markup language. There are 6 main relations and their inverses in Timebank: before, ibefore, includes, begins, ends and simultaneous. Simultaneous is the most prevalent event-event relation in the corpus.

Solely for comparison with Mani et al. (2006), I also add their 73 document Opinion Corpus to create a larger dataset called the OTC. I present both Timebank and OTC results so future work can compare against either.

This chapter describes work that classifies the relations between events, making use of relations between events and times, and between the times themselves to help inform the decisions.

4.2.2 Stage One: Learning Event Attributes

The task in Stage One is to learn the five temporal attributes associated with events as tagged in the Timebank Corpus. (1) Tense and (2) grammatical aspect are necessary in any approach to temporal ordering as they define both temporal location and the shape of the event. (3) Modality and (4) polarity indicate hypothetical or non-occurring situations, and finally, (5) event class is the type of event (e.g., process, state, etc.). The event class has 7 values in Timebank, but this supervised feature-based approach is compatible with other class divisions as well. The range of values for each event attribute is as follows, also found in Pustejovsky et al. (2003):
CHAPTER 4. LEARNING TO ORDER EVENTS

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>tense</td>
<td>none, present, past, future</td>
</tr>
<tr>
<td>aspect</td>
<td>none, prog, perfect, prog_perfect</td>
</tr>
<tr>
<td>class</td>
<td>report, aspectual, state, I_state</td>
</tr>
<tr>
<td></td>
<td>I_action, perception, occurrence</td>
</tr>
<tr>
<td>modality</td>
<td>none, to, should, would, could, can, might</td>
</tr>
<tr>
<td>polarity</td>
<td>positive, negative</td>
</tr>
</tbody>
</table>

Let the following sentence serve as an example of an event and how its attributes affect interpretation:

John had been living here for quite some time.

The event in this sentence is had been living, and its syntactic markers contain important information about its position in time. The tense is past, the grammatical aspect is perfect, and its event class is an occurrence. It has no modality or polarity. The tense and aspect attributes describe the temporal shape of the event and can help in reasoning about the relative position of multiple events (Moens and Steedman, 1988). For instance, if this example is followed by a present tense verb, I can conclude that the living event occurred before it. A past perfect verb followed by a present verb is a strong indicator that the first temporally occurred before the second, and the next section will build a classifier to learn these clues.

Machine Learning Classification

I use a supervised learning approach to identify each of the five event attribute values. I experimented with Naive Bayes, Maximum Entropy, and SVM classifiers, but found Naive Bayes to perform as well or slightly better than the others on this stage one. The small size of the Timebank Corpus may be a factor in the generative approach coming out ahead. The results in this chapter are from Naive Bayes with Laplace smoothing.

The features I used on this stage include part of speech tags of neighboring words (two before the event) and the event word itself, lemmas of the event words, WordNet
Table 4.1: Features selected for learning each temporal attribute. POS-2 is two tokens before the event.

<table>
<thead>
<tr>
<th>Feature</th>
<th>POS-2-event, POS-1-event, POS-of-event, have_word, be_word</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect</td>
<td>POS-of-event, modal_word, be_word</td>
</tr>
<tr>
<td>class</td>
<td>synset</td>
</tr>
<tr>
<td>modality</td>
<td>none</td>
</tr>
<tr>
<td>polarity</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 4.2: Accuracy of the Stage One classifiers.

<table>
<thead>
<tr>
<th>Event Attribute Accuracy</th>
<th>tense</th>
<th>aspect</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>52.21</td>
<td>84.34</td>
<td>54.21</td>
</tr>
<tr>
<td>Accuracy</td>
<td>88.28</td>
<td>94.24</td>
<td>75.2</td>
</tr>
<tr>
<td>Baseline (OTC)</td>
<td>48.52</td>
<td>86.68</td>
<td>59.39</td>
</tr>
<tr>
<td>Accuracy (OTC)</td>
<td>87.46</td>
<td>88.15</td>
<td>76.1</td>
</tr>
</tbody>
</table>

synsets, and the appearance of auxiliaries and modals before the event. This latter set included all derivations of *be* and *have* auxiliaries, modal words (e.g., *may*, *might*, etc.), and the presence/absence of *not*. I performed feature selection (simple ablation experiments) on this list of features, learning a different set of features for each of the five attributes. The list of selected features for each is shown in figure 4.1.

*Modality* and *polarity* did not select any features because their majority class baselines were so high (98%) that learning these attributes does not improve results. A deeper analysis of event interaction would require a modal analysis, but it seems that a newswire domain does not provide great variation in modalities. Consequently, modality and polarity are not used in Stage Two. The classifiers’ accuracy on tense, aspect and class are shown in figure 4.2 with majority class baselines. Tense classification achieves 36% absolute improvement, aspect 10% and class 21%. Performance on the OTC set is similar, although aspect is slightly lower. These imperfect predictions are then passed to Stage Two.
4.2.3 Stage Two: Event-Event Features

The task in this stage is to choose the temporal relation between two events, given the pair of events. I assume that the events have been extracted and that there exists some relation between them (using the Timebank annotations); the task is to predict the correct relation. The Timebank Corpus uses relations that are based on Allen’s set of thirteen (Allen, 1984). Six of the relations are inverses of the other six, and so I condense the set to before, ibefore, includes, begins, ends and simultaneous. I map the thirteenth identity into simultaneous. One oddity is that Timebank includes both during and included by relations, but during does not appear in Timebank documentation. While I don’t know how previous work handles this, I condense during into included by (invert to includes).

Features

The types of features used in my supervised classifier are categorized and described below. Information about the tense, aspect, and event class is used from the output of Stage One’s classifiers.

**Event Specific:** The five temporal attributes from Stage One are used for each event in the pair, as well as the event strings, lemmas and WordNet synsets. Mani et al. (2007) added two other features from these, indicators if the events agree on tense and aspect. I add a third, event class agreement. Further, to capture the dependency between events in a discourse, I create new bigram features of tense, aspect and class (e.g., “present past” if the first event is in the present, and the second is past).

**Part of Speech:** For each event, I include the Penn Treebank POS tag of the event, the tags for the two tokens preceding, and one token following. I use the Stanford Parser\(^1\) to extract them. I also extend previous work and create bigram POS features of the event and the token before it, as well as the bigram POS of the first event and the second event.

\(^1\)http://nlp.stanford.edu/software/lex-parser.shtml
Event-Event Syntactic Properties: A phrase P is said to dominate another phrase Q if Q is a daughter node of P in the syntactic parse tree. I leverage the syntactic output of the parser to create the dominance feature for intra-sentential events. It is either on or off, depending on the two events’ syntactic dominance. Lapata and Lascarides (2006) used a similar feature for subordinate phrases and an indicator before for textual event ordering. I adopt these features and also add a same-sentence indicator if the events appear in the same sentence.

Prepositional Phrase: Since preposition heads are often indicators of temporal class, I created a new feature indicating when an event is part of a prepositional phrase. The feature’s values range over 34 English prepositions. Combined with event dominance (above), these two features capture direct intra-sentential relationships. To my knowledge, this is the first use of this feature in temporal ordering.

Temporal Discourse: Seeing tense as a type of anaphora, it is a natural conclusion that the relationship between two events becomes stronger as the textual distance draws closer. Because of this, I adopted the view that intra-sentential events are generated from a different distribution than inter-sentential events. I therefore train two models during learning, one for events in the same sentence, and the other for events crossing sentence boundaries. It essentially splits the data on the same-sentence feature. As I will show, this turned out to be a very useful feature. It is called the split approach in the next section.

Example (require, compromise):

“Their solution required a compromise…”

Features

(lemma1: require) (lemma2: compromise) (dominates: yes)
(tense-bigram: past-none) (aspect-bigram: none-none)
(tense-match: no) (aspect-match: yes) (before: yes) (same-sent: yes)
<table>
<thead>
<tr>
<th>Timebank Corpus</th>
<th>Gold</th>
<th>Auto</th>
<th>Auto-Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>37.22</td>
<td>37.22</td>
<td>46.58</td>
</tr>
<tr>
<td>Mani</td>
<td>50.97</td>
<td>50.19</td>
<td>53.42</td>
</tr>
<tr>
<td>Mani+Lapata</td>
<td>52.29</td>
<td>51.57</td>
<td>55.10</td>
</tr>
<tr>
<td>All+New</td>
<td>60.45</td>
<td>59.13</td>
<td>59.43</td>
</tr>
</tbody>
</table>

**Mani** stage one attributes, tense/aspect-match, event strings

**Lapata** dominance, before, lemma, synset

**New** prep-phrases, same-sent, class-match, POS uni/bigrams, tense/aspect/class-bigrams

Table 4.3: Incremental accuracy by adding features.

### 4.2.4 Evaluation and Results

All results are from a 10-fold cross validation (as in previous work) using SVM (Chang and Lin, 2001). I also evaluated Naive Bayes and Maximum Entropy. Naive Bayes (NB) returned similar results to SVM and I present feature selection results from NB to compare the added value of my new features.

The input to Stage Two is a list of pairs of events; the task is to classify each according to one of six temporal relations. Four sets of results are shown in figure 4.3. *Mani, Mani+Lapata* and *All+New* correspond to performance on features as listed in the figure. The three table columns indicate how a gold-standard Stage One (*Gold*) compares against imperfect guesses (*Auto*) and the guesses with split distributions (*Auto-Split*).

A clear improvement is seen in each row, indicating that the new features provide significant improvement over previous work. A decrease in performance is seen between columns *gold* and *auto*, as expected, because imperfect data is introduced, however, the drop is manageable. The *auto-split* distributions make significant gains for the Mani and Lapata features, but less when all new features are involved. The highest fully-automatic accuracy on Timebank is 59.43\%, a 4.3\% gain from the new features. I also report 67.57\% *gold* and 65.48\% *auto-split* on the OTC dataset to compare against Mani’s reported hand-tagged features of 62.5\%, a gain of 3\% with automatic features.
Table 4.4: Top 5 features as added in feature selection w/ Naive Bayes, with their percentage improvement.

<table>
<thead>
<tr>
<th>Same Sentence</th>
<th>Diff Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS-1 Ev1</td>
<td>Tense Pair</td>
</tr>
<tr>
<td>2.5%</td>
<td>1.6%</td>
</tr>
<tr>
<td>POS Bigram Ev1</td>
<td>Aspect Ev1</td>
</tr>
<tr>
<td>3.5%</td>
<td>0.5%</td>
</tr>
<tr>
<td>Preposition Ev1</td>
<td>POS Bigram</td>
</tr>
<tr>
<td>2.0%</td>
<td>0.2%</td>
</tr>
<tr>
<td>Tense Ev2</td>
<td>POS-1 Ev2</td>
</tr>
<tr>
<td>0.7%</td>
<td>0.3%</td>
</tr>
<tr>
<td>Preposition Ev2</td>
<td>Word EV2</td>
</tr>
<tr>
<td>0.6%</td>
<td>0.2%</td>
</tr>
</tbody>
</table>

4.2.5 Discussion

Previous work on OTC achieved classification accuracy of 62.5%, but this result was based on “perfect data” from human annotators. A low number from good data is at first disappointing, however, I show that performance can be improved through more linguistic features and by isolating the distinct tasks of ordering inter-sentential and intra-sentential events.

My classifier shows a clear improvement over previous work. The features that capture dependencies between the events, rather than isolated features provide the greatest utility. Also, the impact of imperfect temporal data is surprisingly minimal. Using Stage One’s results instead of gold values hurts performance by less than 1.4%. This suggests that much of the value of the hand-coded information can be achieved via automatic approaches. Stage One’s event class shows room for improvement, yet the negative impact on Event-Event relationships is manageable. It is conceivable that more advanced features would better classify the event class, but improvement on the event-event task would be slight.

Finally, it is important to note the difference in classifying events in the same sentence vs. cross-boundary. Splitting the 3345 pairs of corpus events into two separate training sets makes the training data more sparse, but I still see a performance improvement when using Mani/Lapata features. Figure 4.4 gives a hint to the difference in distributions as the best features of each task are very different. Intra-sentence events rely on syntax cues (e.g., preposition phrases and POS), while inter-sentence events use tense and aspect. However, the differences are minimized as more advanced
features are added. The final row in figure 4.3 shows minimal split improvement.

4.2.6 Pairwise Events Conclusion

I have described a two-stage machine learning approach to event-event temporal relation classification. I have shown that imperfect event attributes can be used effectively, that a range of event-event dependency features provide added utility to a classifier, and that events within the same sentence have distinct characteristics from those across sentence boundaries. This fully automatic raw text approach achieves a 3% improvement over previous work based on perfect human tagged features. Further, the Stage One results may prove useful to other NLP tasks that desire tense and aspect features independent of my event ordering task, such as event coreference and document classification.

4.3 Jointly Combining Implicit Constraints

Most work on event-event ordering has focused on improving classifiers for pairwise decisions, ignoring obvious contradictions in the global space of events when misclassifications occur. The previous section advanced the state-of-the-art in this area, but suffers from the same drawbacks. A global framework to repair these event ordering mistakes had not yet been explored until my publication of the global model described in this section (Chambers and Jurafsky, 2008a).

This chapter addresses three main factors involved in a global framework: the global optimization algorithm, the constraints that are relevant to the task, and the level of connectedness across pairwise decisions. I employ Integer Linear Programming to address the first factor, drawing from related work in paragraph ordering (Bramsen et al., 2006). After finding minimal gain with the initial model, I explore reasons for and solutions to the remaining two factors through temporal reasoning and transitivity rule expansion. I analyze the connectivity of the Timebank Corpus and show how textual events can be indirectly connected through a time normalization algorithm that automatically discovers relations between time expressions (e.g.,
today and tomorrow have a clear after relationship that is not always labeled in the corpus). I show how this increased connectivity is essential for a global model to improve performance.

I present three progressive evaluations of a global model on the Timebank Corpus, showing a 3.6% gain in accuracy over its original set of relations, and an 81% increase in training data size from previous work. In addition, I present the first results on Timebank that include an unknown relation, establishing a benchmark for performance on the full task of document ordering.

### 4.3.1 The Global Model

My initial model has two components: (1) a pairwise classifier between events, and (2) a global constraint satisfaction layer that maximizes the confidence scores from the classifier. The first is based on my pairwise event-event classifier from the previous section (Mani et al., 2007; Chambers et al., 2007) and the second is a novel contribution to global event order classification.

**Pairwise Classification**

Classifying the relation between two events is the basis of my global model. A soft classification with confidence scores is important for the global maximization step that is described in the next section. I use the support vector machine (SVM) classifiers from the previous section and use the probabilities from these pairwise SVM decisions as my confidence scores. These scores are then used to choose an optimal global ordering.

Table 4.5 summarizes the features from the previous event-event section that I use here. They vary from POS tags and lexical features surrounding the event, to syntactic dominance, to whether or not the events share the same tense, grammatical aspect, or aspectual class. These features are the highest performing set on the basic 6-way classification of Timebank.

For the purposes of this comparative study of global constraints, I use Timebank’s gold labeled event attributes.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word*</td>
<td>The text of the event</td>
</tr>
<tr>
<td>Lemma*</td>
<td>The lemmatized head word</td>
</tr>
<tr>
<td>Synset*</td>
<td>The WordNet synset of head word</td>
</tr>
<tr>
<td>POS*</td>
<td>4 POS tags, 3 before, and 1 event</td>
</tr>
<tr>
<td>POS bigram*</td>
<td>The POS bigram of the event and its preceding tag</td>
</tr>
<tr>
<td>Prep*</td>
<td>Preposition lexeme, if in a prepositional phrase</td>
</tr>
<tr>
<td>Tense*</td>
<td>The event’s tense</td>
</tr>
<tr>
<td>Aspect*</td>
<td>The event’s grammatical aspect</td>
</tr>
<tr>
<td>Modal*</td>
<td>The modality of the event</td>
</tr>
<tr>
<td>Polarity*</td>
<td>Positive or negative</td>
</tr>
<tr>
<td>Class*</td>
<td>The aspecual class of the event</td>
</tr>
<tr>
<td>Tense Pair</td>
<td>The two concatenated tenses</td>
</tr>
<tr>
<td>Aspect Pair</td>
<td>The two concatenated aspects</td>
</tr>
<tr>
<td>Class Pair</td>
<td>The two concatenated classes</td>
</tr>
<tr>
<td>POS Pair</td>
<td>The two concatenated POS tags</td>
</tr>
<tr>
<td>Tense Match</td>
<td>true if the events have the same tense</td>
</tr>
<tr>
<td>Aspect Match</td>
<td>true if the events have the same aspect</td>
</tr>
<tr>
<td>Class Match</td>
<td>true if the events have the same class</td>
</tr>
<tr>
<td>Dominates</td>
<td>true if the first event syntactically dominates the second</td>
</tr>
<tr>
<td>Text Order</td>
<td>true if the first event occurs first in the document</td>
</tr>
<tr>
<td>Entity Match</td>
<td>true if they share an entity as an argument</td>
</tr>
<tr>
<td>Same Sent</td>
<td>true if both events are in the same sentence</td>
</tr>
</tbody>
</table>

Table 4.5: The features to learn temporal relations between two events. Asterisks (*) indicate features that are duplicated, one for each of the two events.

**Global Constraints**

Pairwise classifiers can make contradictory classifications due to their inability to consider other decisions. For instance, the following three decisions are in conflict:

A before B
B before C
A after C

Transitivity is not taken into account. In fact, there are several ways to resolve the conflict in this example. Given confidence scores (or probabilities) for each possible
relation between the three pairs, I can compute an optimal label assignment (optimal over the imperfect scores from a classifier). Different scores can lead to different conflict resolutions. Table 4.6 shows two resolutions given different sets of scores. The first chooses before for all three relations, while the second chooses after.

Bramsen et al. (2006) presented a variety of approaches to using transitivity constraints to help inform pairwise decisions. They found that Integer Linear Programming (ILP) performed the best on a paragraph ordering task, consistent with its property of being able to find the optimal solution for a set of constraints. Other approaches are variations on a greedy strategy of adding pairs of events one at a time, ordered by their confidence. These can lead to suboptimal configurations, although they are guaranteed to find a solution. Mani et al. (2007) hypothetically proposed one of these greedy strategies, but did not implement such a system. I implemented both a greedy best-first strategy and ILP, but found ILP to outperform the greedy approach.

My Integer Linear Programming framework uses the following objective function:

$$\max \sum_i \sum_j p_{ij} x_{ij} \tag{4.1}$$
with added constraints:

\[
\forall i \forall j \ x_{ij} \in \{0, 1\} \quad (4.2)
\]
\[
\forall i \ x_{i1} + x_{i2} + ... + x_{im} = 1 \quad (4.3)
\]

where \(x_{ij}\) represents the ith pair of events classified as the jth relation of m relations. Thus, each pair of events generates m variables. Given n pairs of events, there are \(n \times m\) variables. \(p_{ij}\) is the probability of classifying pair i with relation j. Equation (4.2) (the first constraint) simply says that each variable must be 0 or 1. Equation (4.3) contains m variables for a single pair of events i representing its m possible relations. It states that one relation must be set to 1 and the rest to 0. In other words, a pair of events cannot have two relations at the same time. Finally, a transitivity constraint is added for all connected pairs \(i, j, k\), for each transitivity condition that infers relation c given a and b:

\[
x_{ia} + x_{jb} - x_{kc} \leq 1 \quad (4.4)
\]

I generated the set of constraints for each document and used lpsolve\(^2\) to solve the ILP constraint problem.

The transitivity constraints are only effective if the available pairwise decisions constitute a connected graph. If pairs of events are disconnected, then transitivity makes little to no contribution because these constraints are only applicable to connected chains of events.

**Transitive Closure**

In order to connect the event graph, I draw on work from Mani et al. (2007) and apply transitive closure to the documents. Transitive closure was first proposed not to address the problem of connected event graphs, but rather to expand the size of training data for relations such as before. Timebank is a relatively small corpus with few examples of each relation. One way of expand the training set is through

\(^2\)http://sourceforge.net/projects/lpsolve
Total Event-Event Relations After Closure

<table>
<thead>
<tr>
<th></th>
<th>before</th>
<th>after</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timebank</td>
<td>592</td>
<td>656</td>
</tr>
<tr>
<td>+ closure</td>
<td>3919</td>
<td>3405</td>
</tr>
</tbody>
</table>

Table 4.7: The number of event-event relations after transitive closure.

transitive rules. A few rules are given here:

\[
A \text{ simultaneous } B \land A \text{ before } C \rightarrow B \text{ before } C \\
A \text{ includes } B \land A \text{ before } C \rightarrow B \text{ before } C \\
A \text{ before } B \land A \text{ ends } C \rightarrow B \text{ after } C
\]

While the original motivation was to expand the training size of tagged relations, this approach also creates new connections in the graph, replacing previously unla-
beled event pairs with their true relations. I adopted this approach and closed the original set of 12 relations to help connect the global constraint model.

Initial Experiment

The first evaluation of my global temporal model is on the Timebank Corpus over the labeled relations before and after. I merged ibefore and iafter into these two relations as well, ignoring all others. I use this task as a reduced evaluation to study the specific contribution of global constraints. I also chose this strict ordering task because it is well defined from a human understanding perspective. Snow et al. (2008) shows that average internet users can make before/after decisions with very high confidence, although the distinction with an unknown relation is not as clear. An evaluation including unknown (or vague as in TempEval) is presented later.

I expanded the corpus (prior to selecting the before/after relations) using transitive closure over all 12 relations as described above. Table 4.7 shows the increase in data size. The number of before and after relations increase by a factor of six.

I trained and tested the system with 10-fold cross validation and micro-averaged accuracies. The folds were randomly generated to separate the 186 files into 10 folds.
Comparative Results

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timebank Pairwise</td>
<td>66.8%</td>
</tr>
<tr>
<td>Global Model</td>
<td>66.8%</td>
</tr>
</tbody>
</table>

Table 4.8: Using the base Timebank annotated tags for testing, accuracy on before/after tags in the two models.

(18 or 19 files per fold). The same 10-way split is used for all the evaluations. I used libsvm\(^3\) to implement the event-event pairwise SVM classifiers.

Table 4.8 shows the results from my ILP model with transitivity constraints. The first row is the baseline pairwise classification trained and tested on the original Timebank relations. The second row gives performance with ILP. The model shows no improvement. The global ILP constraints did affect local decisions, changing 175 of them (out of 7324), but the changes cancelled out and had no affect on overall accuracy.

Loosely Connected Graph

Why didn’t a global model help? The problem lies in the graph structure of Timebank’s annotated relations. The Timebank annotators were not required to annotate relations between any particular pair of events. Instead, they were instructed to annotate what seemed appropriate due to the almost insurmountable task of annotating all pairs of events. A modest-sized document of 30 events, for example, would contain \(\binom{30}{2} = 435\) possible pairs. Annotators thus marked relations which they deemed fit, most likely between obvious and critical relations to the understanding of the article. The vast majority of possible relations are untagged, thus leaving a large set of unlabeled (and disconnected) unknown relations.

Figure 4.1 graphically shows all relations that are annotated between events and time expressions in one of the shorter Timebank documents. Nodes represent events and times (event nodes start with the letter ‘e’, times with ‘t’), and edges represent

\(^3\)http://www.csie.ntu.edu.tw/~cjlin/libsvm
temporal relations. Solid lines indicate hand annotations, and dotted lines indicate new rules from transitive closure (only one, from event \(e_4\) to time \(t_{14}\)). As can be seen, the graph is largely disconnected and a global model contributes little information since transitivity constraints cannot apply.

![Timebank Annotation of wsj_0551](image)

Figure 4.1: Annotated relations in document wsj_0551.

The large amount of unlabeled relations in the corpus presents several problems. First, building a classifier for these \textit{unknown} relations is easily overwhelmed by the huge training set. Second, many of the untagged pairs have non-\textit{unknown} ordering relations between them, but were missed by the annotators. This point is critical because one cannot filter this noise when training an \textit{unknown} classifier. The noise problem will appear later and will be discussed in my final experiment. Finally, the space of annotated events is very loosely connected and global constraints cannot assist local decisions if the graph is not connected. The results of this first experiment illustrate this latter problem.

Bethard et al. (2007) strengthen the claim that many of Timebank’s untagged relations should not be left unlabeled. They performed an independent annotation of 129 of Timebank’s 186 documents, tagging all events in verb-clause relationships. They found over 600 valid \textit{before/after} relations that are untagged in Timebank, on
average three per document. One must assume that if these nearby verb-clause event
pairs were missed by the annotators, the much larger number of pairs that cross
sentence boundaries were also missed.

The next model thus attempts to fill in some of the gaps and further connect the
event graph by using two types of knowledge. The first is by integrating Bethard’s
data, and the second is to perform temporal reasoning over the document’s time
expressions (e.g., yesterday or january 1999).

4.3.2 A Global Model With Time

My initial model contained two components: (1) a pairwise classifier between events,
and (2) a global constraint satisfaction layer. However, due to the sparseness in
the event graph, I now introduce a third component addressing connectivity: (3) a
temporal reasoning component to inter-connect the global graph and assist in training
data expansion.

One important aspect of transitive closure includes the event-time and time-time
relations during closure, not just the event-event links. Starting with 5,947 different
types of relations, transitive rules increase the dataset to approximately 12,000. How-
ever, this increase wasn’t enough to be effective in global reasoning. To illustrate the
sparsity that still remains, if each document was a fully connected graph of events,
Timebank would contain close to 160,000 relations\(^4\), more than a 13-fold increase.

More data is needed to enrich the Timebank event graph. Two types of informa-
tion can help: (1) more event-event relations, and (2) a separate type of information
to indirectly connect the events: event-X-event. I incorporate the new annotations
from Bethard et al. (2007) to address (1) and introduce a new temporal reasoning pro-
cedure to address (2). The following section describes this novel approach to adding
time expression information to further connect the graph.

\(^4\)Sum over the \# of events \(n_d\) in each document \(d\), \(\binom{n_d}{2}\)
Time-Time Information

As described above, I use event-time relations to produce the transitive closure, as well as annotated time-time relations. It is unclear if Mani et al. (2007) used these latter relations in their work.

However, I also add new time-time links that are deduced from the logical time intervals that they describe. Time expressions can be resolved to time intervals with some accuracy through simple rules. New time-time relations can then be added to the space of events through time stamp comparisons. Take this newswire example:

*The Financial Times 100-share index shed 47.3 points to close at 2082.1, down 4.5% from the previous Friday, and 6.8% from Oct. 13, when Wall Street’s plunge helped spark the current weakness in London.*

The first two expressions (‘previous Friday’ and ‘Oct. 13’) are in a clear before relationship that Timebank annotators captured. The ‘current’ expression, is correctly tagged with the PRESENT_REF attribute to refer to the document’s timestamp. Both ‘previous Friday’ and ‘Oct. 13’ should thus be tagged as being before this expression. However, the annotators did not tag either of these two before relations, and so my timestamp resolution procedure fills in these gaps. This is a common example of two expressions that were not tagged by the annotators, yet are in a clear temporal relationship.

I use Timebank’s gold standard TimeML annotations to extract the dates and times from the time expressions. In addition, those marked as PRESENT_REF are resolved to the document timestamp. Time intervals that are strictly before or after each other are thus labeled and added to the space of events. I then create new before relations based on the deterministic procedure given in figure 4.2. As in the example given above, the phrases ‘Oct 13’ and ‘previous Friday’ are sent to this procedure and a before link is automatically added. All other time-time orderings not including the before relation are ignored (i.e. includes is not created, although could be with minor changes).
function isBefore(interval1, interval2) {
    if interval1.year < interval2.year
        return true
    if interval1.year == interval2.year
        if interval1.month < interval2.month
            return true
        if interval1.month == interval2.month
            if interval1.day < interval2.day
                return true
        end
    end
    return false
end

Figure 4.2: The straightforward procedure that compares two time intervals.

This new time-time knowledge is used in two separate stages of my model. The first is just prior to transitive closure, enabling a larger expansion of the tagged relations set and reduce the noise in the unknown set. The second is in the constraint satisfaction stage where I add the automatically computed time-time relations (with the gold event-time relations) to the global graph to help correct local event-event mistakes.

**Temporal Reasoning Experiment**

My second evaluation continues the use of the two-way classification task with before and after to explore the contribution of closure, time normalization, and global constraints.

I augmented the corpus with the labeled relations from Bethard et al. (2007) and added the automatically created time-time relations as described in section 4.3.2. I then expanded the corpus using transitive closure. Table 4.9 shows the progressive data size increase as I incrementally add each to the closure algorithm.

The time-time generation component automatically added 2459 new before and after time-time relations into the 186 Timebank documents. This is in comparison
CHAPTER 4. LEARNING TO ORDER EVENTS

Total Event-Event Relations After Closure

<table>
<thead>
<tr>
<th></th>
<th>before</th>
<th>after</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timebank</td>
<td>3919</td>
<td>3405</td>
</tr>
<tr>
<td>+ time-time</td>
<td>5604</td>
<td>5118</td>
</tr>
<tr>
<td>+ time/bethard</td>
<td>7111</td>
<td>6170</td>
</tr>
</tbody>
</table>

Table 4.9: The number of event-event before and after relations after transitive closure on each dataset.

Comparative Results with Closure

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timebank Pairwise</td>
<td>66.8%</td>
</tr>
<tr>
<td>Global Model</td>
<td>66.8%</td>
</tr>
<tr>
<td>Global + time/bethard</td>
<td>70.4%</td>
</tr>
</tbody>
</table>

Table 4.10: Using the base Timebank annotated tags for testing, the increase in accuracy on before/after tags.

to only 157 relations that the human annotators tagged, less than 1 per document on average. The second row of table 4.9 shows the drastic effect that these time-time relations have on the number of available event-event relations for training and testing. Adding both Bethard’s data and the time-time data increases the training set’s size by 81% over closure without it.

Using this expanded dataset for training, I again performed 10-fold cross validation with micro-averaged accuracies. However, each fold tested only on the transitively closed Timebank data (the first row of table 4.9). In other words, the Bethard data is not included in the test sets since most other work does not evaluate on this set. In addition, accuracy numbers may be inflated on the Bethard data as they focus on a specific language construct. The training set, on the other hand, uses all available data (the third row of table 4.9) including the Bethard data as well as my new time-time links for closure.

Table 4.10 shows the results from this new model. The first row is the baseline pairwise classification trained and tested on the original Timebank labeled relations only. My model improves on this baseline by 3.6% absolute. This improvement is
Discussion

To further illustrate why my model now improves local decisions, I continue my previous graph example. The actual text for the graph in figure 4.1 is shown here:

\[\text{docstamp: } 10/30/89 \ (t14)\]

\[\text{Trustcorp Inc. will become(e1) Society Bank \& Trust when its merger(e3) is completed(e4) with Society Corp. of Cleveland, the bank said(e5). Society Corp., which is also a bank, agreed(e6) in June(t15) to buy(e8) Trustcorp for 12.4 million shares of stock with a market value of about }\$450\text{ million. The transaction(e9) is expected(e10) to close(e2) around year end(t17).}\]

The automatic time normalizer computes and adds three new time-time relations, two connecting t15 and t17 with the document timestamp, and one connecting t15 and t17 together. These are not otherwise tagged in the corpus. Figure 4.3 shows the augmented document. The double-line arrows indicate the three new time-time relations and the dotted edges are the new relations added by my transitive closure procedure. Most critical to my global framework, three of the new edges are event-event relations that help to expand the training data’s size. When this document is used in testing (rather than training), these new edges help inform my transitive rules during classification within the ILP framework.

Even with this added information, disconnected segments of the graph are still apparent even in this example. However, the 3.6% performance gain encourages us to move to the final full task.

4.3.3 Final Experiment with Unknowns

The final evaluation expands the set of relations to include unlabeled relations and tests on the entire dataset available to us. The following is now a classification task between the three relations: \textit{before}, \textit{after}, and \textit{unknown}. 
I duplicated the previous evaluation by adding the labeled relations from Bethard et al. (2007) and my automatically created time-time relations. I then expanded this dataset using transitive closure. Unlike the previous evaluation, I also use this entire dataset for testing, not just for training. Thus, all event-event relations in Bethard as well as Timebank are used to expand the dataset with transitive closure and are used in training and testing. I wanted to fully evaluate document performance on every possible event-event relation that logically follows from the data.

As before, I converted IBefore and IAfter into before and after respectively, while all other relations are reduced to unknown. This relation set coincides with TempEval-07’s core three relations (although they use vague instead of unknown).

Rather than include all unlabeled pairs in the unknown set, I only include the unlabeled pairs that span at most one sentence boundary. In other words, events in adjacent sentences are included in the unknown set if they were not tagged by the Timebank annotators. The intuition is that annotators are more likely to label nearby events, and so events in adjacent sentences are more likely to be actual unknown relations if they are unlabeled. The distant events in the text were likely overlooked by convenience, not because they truly constituted an unknown relationship.

Figure 4.3: Before and after time-time links with closure.
CHAPTER 4. LEARNING TO ORDER EVENTS

Classification Accuracy

<table>
<thead>
<tr>
<th>% unk</th>
<th>Base (Just Pairs)</th>
<th>Global Model</th>
<th>Global+Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>72.0%</td>
<td>72.2%</td>
<td>74.0%</td>
</tr>
<tr>
<td>1</td>
<td>69.4%</td>
<td>69.5%</td>
<td>71.3%</td>
</tr>
<tr>
<td>3</td>
<td>65.5%</td>
<td>65.6%</td>
<td>67.1%</td>
</tr>
<tr>
<td>5</td>
<td>63.7%</td>
<td>63.8%</td>
<td>65.3%</td>
</tr>
<tr>
<td>7</td>
<td>61.2%</td>
<td>61.6%</td>
<td>62.8%</td>
</tr>
<tr>
<td>9</td>
<td>59.3%</td>
<td>59.5%</td>
<td>60.6%</td>
</tr>
<tr>
<td>11</td>
<td>58.1%</td>
<td>58.4%</td>
<td>59.4%</td>
</tr>
<tr>
<td>13</td>
<td>57.1%</td>
<td>57.1%</td>
<td>58.1%</td>
</tr>
</tbody>
</table>

Table 4.11: Overall accuracy when training with different percentages of unknown relations included. 13% of unknowns is about equal to the number of befores.

The set of possible sentence-adjacent unknown relations is very large (approximately 50000 unknown compared to 7000 before), and so I randomly select a percentage of these relations for each evaluation. I used the same SVM approach with the features described in section 4.3.1.

Results

Results are presented in table 4.11. The rows in the table are different training/testing runs on varying sizes of unknown training data. There are three columns with accuracy results of increasing complexity. The first, base, are results from pairwise classification decisions over Timebank and Bethard with no global model. The second, global, are results from the Integer Linear Programming global constraints, using the pairwise confidence scores from the base evaluation. Finally, the global+time column shows the ILP results when all event-time, time-time, and automatically induced time-time relations are included in the global graph.

The ILP approach does not alone improve performance on the event-event tagging task, but adding the time expression relations greatly increases the global constraint results. This is consistent with the results from our first two experiments. The evaluation with 1% of the unknown tags shows an almost 2% improvement in accuracy. The gain becomes smaller as the unknown set increases in size (1.0% gain with 13%
### Base Pairwise Classification

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>61.4</td>
<td>55.4</td>
<td>58.2</td>
</tr>
<tr>
<td>after</td>
<td>57.6</td>
<td>53.1</td>
<td>55.3</td>
</tr>
<tr>
<td>unk</td>
<td>53.0</td>
<td>62.8</td>
<td>57.5</td>
</tr>
</tbody>
</table>

### Global+Time Classification

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>63.7 (+2.3)</td>
<td>57.1 (+2.2)</td>
<td>60.2 (+2.0)</td>
</tr>
<tr>
<td>after</td>
<td>60.3 (+2.7)</td>
<td>54.3 (+2.9)</td>
<td>57.1 (+1.8)</td>
</tr>
<tr>
<td>unk</td>
<td>52.0 (-1.0)</td>
<td>62.9 (+0.1)</td>
<td>56.9 (-0.6)</td>
</tr>
</tbody>
</table>

Table 4.12: Precision and Recall for the base pairwise decisions and the global constraints with integrated time information.

*unknown*. *Unknown* relations will tend to be chosen as more weight is given to *unknowns*. When there is a constraint conflict in the global model, *unknown* tends to be chosen because it has no transitive implications. All improvements from base to global+time are statistically significant ($p < 0.000001$, McNemar’s test, 2-tailed).

The first row of table 4.11 corresponds to the results in my second experiment in table 4.10, but shows higher accuracy. The reason is due to my different test sets. This final experiment includes Bethard’s event-event relations in testing. The improved performance suggests that the clausal event-event relations are easier to classify, agreeing with the higher accuracies originally found by Bethard et al. (2007).

Table 4.12 shows the precision, recall, and f-score for the evaluation with 13% *unknowns*. This set was chosen for comparison because it has a similar number of *unknown* labels as *before* labels. I see an increase in precision in both the *before* and *after* decisions by up to 2.7%, an increase in recall up to 2.9%, and an F1 score by as much as 2.0%. The *unknown* relation shows mixed results, possibly due to its noisy behavior as previously discussed in this chapter.
Discussion

My results on the two-way (before/after) task show that explicitly adding implicit temporal constraints and then performing global reasoning results in significant improvements in temporal ordering of events (3.6% absolute over simple pairwise decisions).

Both before and after also showed increases in precision and recall in the three-way evaluation. However, unknown did not parallel this improvement, nor are the increases as dramatic as in the two-way evaluation. I believe this is consistent with the noise that exists in the Timebank corpus for unlabeled relations. Evidence from Bethard’s independent annotations directly point to missing relations, but the dramatic increase in the size of the closure data (81%) from adding a small amount of time-time relations suggests that the problem is widespread. This noise in the unknown relation may be dampening the gains that the two way task illustrates.

This work is also related to the task of event-time classification. While not directly addressed in this chapter, the global methods described within clearly apply to pairwise models of event-time ordering as well.

Further progress in improving global constraints will require new methods to more accurately identify unknown events, as well as new approaches to create implicit constraints over the ordering. I expect such an improved ordering classifier to be used to improve the performance of tasks such as summarization and question answering about the temporal nature of events.

Finally, this complete document event-event classifier can be used to learn the type of generalized event orderings (e.g., arrest is typically before convict) that are needed for narrative schemas. The next section lays out this connection between narrative schemas and an event ordering system.
Chapter 5

Ordering Event Schemas

Chapters 2 and 3 described how to learn rich event structure from unlabeled, unclassified text. Chapter 4 described a series of supervised learning models that identify the temporal ordering between pairs of events in text. I will now merge these two processes and impose a partial ordering over my learned narrative schemas using the temporal classifiers.

As the previous chapter discussed, there are a number of algorithms for determining the temporal relationship between two events (Mani et al., 2006; Lapata and Lascarides, 2006; Chambers et al., 2007), many of them trained on the TimeBank Corpus (Pustejovsky et al., 2003). The currently highest performing of these on raw TimeBank text is my model of temporal labeling just described in the previous Chapter 4. Though the global model is most sophisticated, it requires full temporal reasoning over time expressions to connect the events in a document, and so I only focus on the pairwise supervised classifier here. Further, instead of using all six possible relations between pairs of events (before, immediately-before, included-by, simultaneous, begins and ends), I will instead focus solely on the before relation. The other relations are less relevant to our immediate task of ordering event schemas, and training data for them is quite sparse. I also want higher accuracy in the final ordering, so I will focus instead on the more prevalent before relation. I combine immediately-before with before, and merge the other four relations into an other category. This is different from the unknown category of the previous chapter, but rather
is a catch-all class that includes all known relations that are not before or after.

Using this trained classifier, I will classify all pairs of events in the Gigaword Corpus and observe which pairs of events are consistently classified with a single relation. To my knowledge, this is the first work to use a supervised classifier that collects ordering statistics over a corpus the size of Gigaword. These statistics make up the final ordering decisions in my learned narrative schemas.

5.1 Related Work

Chapter 4 covered in detail previous work on supervised learning of event pairs in context, but I now instead focus on learning general event orderings, such as, arrest is usually before convict. This chapter describes how to use supervised classifiers to make this generalization.

Approaches to knowledge acquisition have effectively used unsupervised and semi-supervised algorithms to learn a type of general event order. (Chklovski and Pantel, 2004) learned pairs of ordered events by using Hearst-style patterns (Hearst, 1992) to extract tokens in happens-before and enablement relationships. They queried a web search engine for hand-built patterns like “to X and then Y” and “Xed and later Yed” and determined significance using mutual information. (Chklovski and Pantel, 2005) then studied how to chain pairs of these happens-before relations together. Their approach can extract repeated precise event pairs, but it has the drawback of being limited to only events that happen to appear in a predefined set of string patterns. The vast majority of before relationships in individual documents do not appear in these patterns.

This chapter therefore extracts similar pairs of events, but with a higher level of recall. Instead of relying on a set of hand-coded patterns, it classifies every pair of events in every document. These counts are then used to order the event pairs that have already been learned for narrative schemas. This is the first algorithm to use a supervised classifier over such a large corpus to collect statistics on event orderings.
5.2 Training a Temporal Classifier

I use the entire Timebank Corpus as supervised training data, condensing the before and immediately-before relations into one before relation. The remaining relations are merged into other.

As discussed in Chapter 4, the vast majority of potential event pairs in Timebank are unlabeled. These are often none relations (events that have no explicit relation) or as is sometimes the case, overlap relations where the two events have no Timebank-defined ordering but overlap in time. Even worse, many events do have an ordering, but they were not tagged by the human annotators. This could be due to the overwhelming task of temporal annotation, or simply because some event orderings are deemed more important than others in understanding the document. I consider all untagged relations as unknown, and merge these into the other category. I experiment with including different amounts of unknown relations by randomly selecting none, half, and all of them in training.

I also increased Timebank’s training size by applying transitivity rules to the hand labeled data, as described in Chapter 4. Since the annotators do not label all possible relations, inferring new relations through transitivity increases the training data size. Transitivity rules have obvious applications within the same relation, such as the following example:

\[
\text{if run BEFORE fall and fall BEFORE injured} \\
\quad \text{then run BEFORE injured}
\]

But there are less obvious rules that mix relation types which can also be applied, greatly increasing the number of relations. The following is one such example:

\[
\text{if run INCLUDES fall and fall SIMULTANEOUS injured} \\
\quad \text{then run INCLUDES injured}
\]

I encoded all possible transitivity rules between all possible relations, and applied them to Timebank. This increases the number of relations from 37519 to 45619, a 22% increase in size. Perhaps more importantly for our task, of all the added relations, the before relation is added the most. I then condensed the resulting relations into before
and other, as described above. I experimented with both the original set of relations in Timebank and this expanded set from transitivity. In contrast to the results in chapter 4 when temporal expressions were included, using transitivity did not help performance. The lack of improvement may be due to poor transitivity additions, as several Timebank documents contain inconsistent labelings. The simultaneous relation is boosted in transitivity quite often, and increases the results of the previous section. However, focusing on the before relation does not have the same improvement. All reported results are thus from training without transitivity.

I also use the same two-stage machine learning architecture from Chapter 4. The first stage uses supervised machine learning to label temporal attributes of events, including tense, grammatical aspect, and aspectual class. This first stage classifier relies on features such as neighboring part of speech tags, neighboring auxiliaries and modals, and WordNet synsets. Instead of solely relying on Naive Bayes, I also experiment with a support vector machine and see minor performance boosts on the test set. These imperfect classifications, combined with other linguistic features, are then used in the second stage to classify the temporal relationship between two events.

5.3 Temporal Classifier in Narrative Chains

This section describes how to apply the above binary before/other classifier to the New York Times section of the Gigaword Corpus (Graff, 2002). This is the same corpus from which chapters 2 and 3 learned narrative schemas. To order the learned sets of events, I will classify each observed pair of events in the context of its document and observe the frequency of before relations.

Let the pair of events arrest and plead serve as an example of this process. These two verbs were learned as part of a criminal prosecution narrative schema, but we do not know in what order they generally occur. I ran the classifier over all occurrences of both verbs in the same document, recording their before/other counts. Figure 5.1 shows four such passages. Three of the four illustrate the stereotypical arrest before plead relation, with one noisy example involving multiple arrest events. I then count
Two Hartford police officers were arrested Friday and charged with beating a Massachusetts man...Lawyers for Ancona and Middleton said their clients would plead not guilty.

(\textit{arrest is before plead})

Pratt, the Cyclones’ second-leading scorer and rebounder last season, was arrested Saturday night after being involved in a wreck. ... This came after Pratt had pleaded guilty Nov. 19 to disorderly conduct.

(\textit{plead is before arrest})

Last May 24, the FBI arrested a fourth suspect, Kevin McCarthy, 19, of Philadelphia. McCarthy, who will be an important witness in Langan’s trial, has agreed to plead guilty.

(\textit{arrest is before plead})

Ms. Woodward, who turned 19 last week, pleaded not guilty to assault charges in February. She has been held without bail at the maximum security Framingham prison since her arrest on Feb. 5.

(\textit{arrest is before plead})

Figure 5.1: Four passages from the NYT section of the Gigaword Corpus containing arrest and plead. The temporal classifier classifies each passage independently, determining if each pair is in a \textit{before} or \textit{other} relation. The second example illustrates a difficult case involving multiple arrest events.

The number of befores and others across the entire corpus.

This classification process over the Gigaword Corpus occurs in two stages: once for the temporal features on each event (tense, grammatical aspect, aspectual class), and once between all pairs of events that share arguments. This allows us to classify the before/other relations between all potential narrative events.

The first stage is trained on Timebank as before, and the second is trained using the approach just described, varying the size of the \textit{unknown} training relations. Each pair of events in a gigaword document that share a coreferring argument is treated as a separate ordering classification task. I count the resulting number of labeled before relations between each verb/dependency pair. Processing the entire corpus produces a database of event pair counts where confidence of two generic events A and B is
<table>
<thead>
<tr>
<th>Event Pair</th>
<th># Before</th>
<th># After</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrest, convict</td>
<td>684</td>
<td>22</td>
</tr>
<tr>
<td>arrest, acquit</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>arrest, plead</td>
<td>132</td>
<td>1</td>
</tr>
<tr>
<td>search, arrest</td>
<td>22</td>
<td>37</td>
</tr>
<tr>
<td>search, sentence</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>search, convict</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>convict, sentence</td>
<td>579</td>
<td>457</td>
</tr>
<tr>
<td>acquit, convict</td>
<td>209</td>
<td>35</td>
</tr>
</tbody>
</table>

Figure 5.2: Before and after counts between pairs of events in the criminal domain: counts of classification decision over the NYT portion of the Gigaword Corpus.

measured by comparing how many before labels have been seen versus their inverted order B and A.

Figure 5.2 shows the before/after counts of event pairs in a criminal domain, the same events in the first schema from figure 1.1. The final pair, acquit and convict, is most interesting. My error analysis on the acquit/convict pair discovered two types of passages that make up the vast majority of acquit before convict classifications. The first is from passages that discuss multiple trials with the same participant, but that occur at different times. Once someone is convicted of a crime, they more often than not are sentenced to prison, and so have less of a chance of being acquitted of future crimes. However, someone who is acquitted can still be convicted in a separate trial on later charges. Figure 5.4 shows three such text examples that were classified as acquit before convict. Learning to detect event relationships across disparate schemas such as these remains for future work.

The second type of passage is more prominent, and involves acquittals from some charges, but convictions on other charges involving the same crime. Figure 5.5 shows several such text passages, all typically mentioned in a single sentence. The English preference seems to be listing the acquittals first, followed by the convictions. The classifier thus chooses a before relation in these cases. It is debatable if this should be a simultaneous relation or not. The fine-grained distinctions between a crime,

---

1Note that I train with the before relation, and so transposing two events is similar to classifying the after relation.
multiple charges on the same crime, and different verdicts on each charge are not captured by this dissertation’s learning algorithms. Future work will need to use intra-sentence clues like these to learn this level of detail.

5.4 Temporal Evaluation

Previous work used human evaluations to judge individual event pair decisions (Chklovskii and Pantel, 2004), but the goal of this dissertation is to order entire schemas, not pairwise events. I want to evaluate the temporal order at a narrative level, across all events within a schema. For narrative schemas to be used for tasks of coherence, among other things, it is desirable to evaluate temporal decisions within a coherence framework. Along these lines, my test set uses actual narrative chains from (coherent) documents, hand labeled for a partial ordering. I evaluate coherence of these true chains against a random ordering. The task is thus deciding which of the two chains is most coherent, the original or the random (baseline 50%)? I generated up to 300 random orderings for each test document, averaging the accuracy across all. This is similar to the ordering evaluations used in summarization (Barzilay and Lee, 2004) and event learning (Manshadi et al., 2008).

The evaluation data is the same 69 documents used in the test set for the narrative cloze evaluations in Chapters 2 and 3. The chain from each document is hand identified and labeled for a partial ordering using only the before relation. I manually ordered the events and all attempts were made to include every before relation that exists in the document, or that could be deduced through transitivity rules. These include before relations that are semantically connected in a temporal sense (e.g., departed before arrived), but also those that are before in temporal order but not necessarily semantically connected in a temporal sense (e.g., departed before laughed). Figure 5.3 shows an example and its full reversal, although the evaluation uses random orderings. Each edge is a distinct before relation and is used in the judgement score.

The coherence score for a partially ordered narrative chain is the sum of all the relations that our classified corpus agrees with, weighted by confidence. If the gigaword
classifications disagree, a negative score is given. Confidence is based on a logarithm scale of the difference between the counts of before and after classifications. Formally, the score is calculated as the following:

\[
\sum_{E:x,y} \begin{cases} 
\log(D(x,y)) & \text{if } x\beta y \text{ and } B(x,y) > B(y,x) \\
-\log(D(x,y)) & \text{if } x\beta y \text{ and } B(y,x) > B(x,y) \\
-\log(D(x,y)) & \text{if } !x\beta y \& !y\beta x \& D(x,y) > 0 \\
0 & \text{otherwise} 
\end{cases}
\]

where \( E \) is the set of all event pairs, \( B(i,j) \) is how many times the classifier classified events \( i \) and \( j \) as before in Gigaword, and \( D(i,j) = |B(i,j) - B(j,i)| \). The relation \( i\beta j \) indicates that \( i \) is temporally before \( j \).

### 5.5 Results

My approach gives higher scores to schema orderings that coincide with the pairwise orderings observed in the Gigaword training data. The results are shown in table 5.1. Of the 69 test event chains, six did not have any strictly ordered (e.g., before/after)
Table 5.1: Results for choosing the correct ordered chain. \textit{At least 10} indicates that there were at least 10 pairs of ordered events in the chain.

relations and were removed from the evaluation. These contained only overlap and vague relations. I generated (up to) 300 random orderings for each of the remaining 63. Final accuracy on predicting the original order is 75.2\%. However, 22 of the 63 had five or fewer pairs of ordered events. The final choice in these cases is dependent on very little evidence of order. Table 5.1 therefore gives results from the test chains based on the richness of their orderings. The table shows results for chains with at least six relations and at least 10 relations, as well as the overall score. As I would hope, the accuracy improves the larger the ordered narrative chain. I achieve 89.0\% accuracy on the 24 documents whose chains are most richly connected with the before relation, rather than chains whose strict order is more ambiguous.

Finally, I found that the classifier trained without any unknown relations resulted in higher recall for before decisions. Perhaps due to data sparsity in the small Timebank Corpus, this produces our best results and is reported above. The best classifier is thus a two-class before/other classifier where the other class is the union of all labeled Timebank relations that are not before or after.

5.6 Error Analysis

This section looks at a few examples of misclassified event pairs and the reasons that cause the errors.

The final event pair in Figure 5.2, acquit and convict, is most interesting in that it contains 209 instances of acquit occurring before convict, but very few of the inverse (35 instances). This is at first counter-intuitive because intuition suggests
that trials end with one or the other verdict, but not both. My error analysis on the
decide/convict pair discovered two types of passages that make up the vast majority
of these decide before convict classifications. The first is from passages that discuss
multiple trials with the same participant, but that occur at different times. Once
someone is convicted of a crime, they more often than not are sentenced to prison,
and so have less of a chance of being acquitted of future crimes. However, someone
who is acquitted can still be convicted in a separate trial on later charges. Figure 5.4
shows three such text examples that were classified as acquit before convict. Learning
to detect event relationships across disparate schemas such as these remains for future
work. Some of these examples describe events with ambiguous modalities, such as
the first example in Figure 5.4. In this case, a man was acquitted, but his potential
conviction is discussed as a possible event that did not occur. Dealing with complex
modalities and interpreting their correct temporal order remains for future work.

The second type of acquit/convict passage is more prominent, and involves acquit-
tals from some charges, but convictions on other charges involving the same crime.
Figure 5.5 shows several such text passages, all typically mentioned in a single sen-
tence. The English preference seems to be listing the acquittals first, followed by the
convictions. The classifier thus chooses a before relation in these cases. It is debatable
if this should be a simultaneous relation or not. The fine-grained distinctions between
a crime, multiple charges on the same crime, and different verdicts on each charge
are not captured by this dissertation’s learning algorithms. Future work will need to
use intra-sentence clues like these to learn this level of detail.

Many errors in temporal ordering arise from the small size of the Timebank Cor-
pus, and the lack of robustness in a supervised classifier trained on it. Let the obituary
example from Figure 2.7 in Chapter 2 serve as an example. The died event has a
directed arrow into the retired event, suggesting that people die before they retire.
In fact, my Timebank-trained classifier labeled (mostly incorrectly) 295 examples of
dying before retiring, and only 1 example of retiring before dying. This is obviously
incorrect, and the reason lies in our small training data. The event die only occurs
twice in Timebank, once in a before relation and once in an after relation. Neither
are with retire, so it is ambiguous. However, the event retire occurs in 6 labeled event
Zamarripa, 41, was acquitted last year. Now, he's suing the Mesa Police Department, claiming he was coerced into making a false confession after taking a polygraph test. The lawsuit details how a man admitted a crime that a jury agreed he didn’t commit. His lawyers said Zamarripa, who now works at a convenience store, was a victim of a carefully orchestrated plan to get him to admit the allegations. “When asked the right questions and told the right lies, under circumstances designed to be coercive, an innocent suspect confessed because he believed he could be convicted by his own silence.”

In two prior trials in 1995, juries acquitted Bly of drug trafficking for allegedly selling cocaine on Morton Street in Mattapan and for his alleged role in a drive-by shooting in Dorchester that wounded a man. Sources said the timing of McLaughlin’s murder helped make Bly a key suspect. McLaughlin was set to begin Bly’s third trial, on carjacking and attempted murder charges, the next day. His colleague, Michael Pomarole, took over Bly’s prosecution later that year and subsequently convicted him. Superior Court Judge Patrick King sentenced him to 10 to 15 years in prison.

A jury in November acquitted Durst of murdering neighbor Morris Black. .... Durst then tampered with evidence of a crime when he tossed the body parts into the bay, according to the indictment. .... In two additional motions filed Friday, DeGuerin also alleged that the evidence-tampering indictment should be tossed out because its wording fails to meet legal requirements and because prosecutors filed the case under the wrong portion of the Texas Criminal Code. If convicted, Durst could be sentenced to up to 10 years in prison on each of the three charges.

Figure 5.4: Passages from the NYT where acquit was classified as before convict. All three are correct classifications involving different trial periods. Two of the three are hypothetical convictions described with modalities occurring in the future.
When the trial is over, whether Simpson is **acquitted** or **convicted** of murder charges, Fuhrman is expected to leave the Los Angeles Police Department and move out of state.

Earlier in the year, Nosair had been **acquitted** of murder but **convicted** of gun possession charges in the killing of Kahane in 1990.

A jury **acquitted** the two teen-agers of attempted murder but **convicted** them of second-degree assault and first-degree attempted assault.

After deliberating for nearly two days, the 12-member jury **convicted** Alexander Blarek, 56, and Frank Pellecchia, 49, on single counts of racketeering conspiracy, money laundering and transporting drug proceeds across state lines. The jury **acquitted** the two men on racketeering charges.

Goetz was **acquitted** of assault and attempted murder in 1987 but was **convicted** on a weapons possession charge.

Four days later, the jury **acquitted** the defendants on charges of conspiring to disrupt the Democratic National Convention, but **convicted** five of them, including Hoffman, of crossing state lines with intent to riot.

Figure 5.5: Passages from the NYT where **acquit** was classified as before **convict**. These examples illustrate that sentence-internal ordering of past tense verbs favors before relations.
However, cited by District of Columbia traffic police in December for driving under the influence of alcohol, Farkas was ordered home and retired.

cited BEFORE retired
driving BEFORE retired

Frederick B. Taylor, 48, also was named a vice chairman and chief investment officer, a new post. He previously held similar responsibilities. Mr. Taylor also was named a director, increasing the board to 22, but is not part of the new office of the chairman. James E. Bacon, 58, executive vice president, who has directed the funds-service group, will retire.

named BEFORE retired
held BEFORE retired
directed BEFORE retired

Figure 5.6: Examples of text and the gold labeled BEFORE relations in the Timebank Corpus for the event retire.

pairs, all of which put it in the after position. Figure 5.6 shows some of these labeled relations. Since the classifier sees such strong evidence that retire is always ordered second, it will prefer that label unless strong syntactic evidence suggests otherwise. 295 instances in the Gigaword Corpus are thus incorrectly labeled, mostly in obituaries, as die before retire. Figure 5.6 shows a few examples where these two occur together. This is a common problem with my approach that needs to be addressed in future work. When two events occur in separate sentences, there is little syntactic evidence, and so the small amount of training data is easily led astray. New sources of information, perhaps from semi-supervised datasets is needed to address this area.

5.7 Discussion

These results indicate that a supervised classifier can help discover general event orders when run over a large corpus of text. The individual contexts within documents may vary widely, but consistencies across them can be easily extracted with the classifier. I found that a coherence application to judge a document’s events can
Don Cook, a foreign correspondent who covered the end of World War II in Europe and then Western Europe’s remarkable postwar recovery for The New York Herald Tribune and The Los Angeles Times, died Tuesday at his home in Philadelphia. He was 74. The cause was a heart attack, his family said. Beginning in 1944, Cook covered the final year of the war, the turbulent early years of the Cold War, the economic transformation of Western Europe and the birth of the Common Market, first for The Herald Tribune and then for The Times, where he retired in 1988 as the European diplomatic correspondent.

Morris B. Zale, founder of the jewelry-store chain Zale Corp., died Wednesday at Presbyterian Hospital in Dallas. He was 93 and lived in Dallas. The cause was pneumonia, said his son Donald. Zale started the company in 1924 as a small jewelry store in Wichita Falls, Texas. By 1986, when the Zale Corporation was acquired in a hostile takeover by Peoples Jewelers of Canada and Swarovski International Holdings AG of Switzerland, Zale had built it into an international company with more than 1,500 stores and $1.2 billion in annual sales. Zale stepped down as president in 1971 and became the chairman. He was appointed chairman emeritus in 1981 and retired in 1987.

John Michael Dunn, a veteran of three wars who rose to become a major general and a military aide to two vice presidents, died on Friday at his home in Arlington, Va. He was 69. Dunn stayed on the vice-presidential staff until early 1974, when he retired from the military and became the principal assistant secretary of commerce for domestic and international business.

Figure 5.7: Examples of obituaries where the retire event is incorrectly classified as after the die event.
be addressed with such counts, and they show promise for future applications in reasoning, causation, and event prediction.

One of the main drawbacks of this algorithm is the dependence on the Timebank Corpus for training the supervised classifier. Given only 186 documents, the classifier is exposed to the main topics in news, such as finance and crime, but lacks most others. The classifier is thus dependent on its topic-general features like tense, aspect, and syntactic positions to make its ordering decisions. Although useful features, as with many NLP applications, the lexicalized features often produce the largest improvement in performance when utilized. Unfortunately, lexicalized models are dependent on a large labeled corpus with a diverse set of tokens. The Timebank Corpus is not large enough to benefit from these features. This is evident in the acquit/convict error analysis of Section 5.3. Those two verbs do not appear together in Timebank, so the classifier strongly decides (arguably, incorrectly) that acquit is before convict based on syntactic properties alone. Anecdotally, the Timebank’s covered topics produce the best schema orderings, and the rest include more noise. Future work will require integrating new unsupervised approaches to temporal relation classification.
Chapter 6

Learning Events for Template-Based Information Extraction

This chapter now turns to the field of Information Extraction (IE) to illustrate how algorithms for learning narrative schemas apply to a mainstream NLP application. As narrative schemas can encode multiple entities, template-based information extraction is the most obvious candidate to benefit from my approaches. A template defines a specific type of event (e.g., a bombing) with a set of semantic roles (or slots) for the typical entities involved in such an event (e.g., perpetrator, target, instrument). In contrast to other IE work, such as relation discovery, that focuses on learning atomic facts (Banko et al., 2007a; Carlson et al., 2010b), templates extract a richer representation of a particular domain, much like narrative schemas. However, unlike relation discovery, most template-based IE approaches assume foreknowledge of the domain’s templates. Very little work addresses how to learn the template structure itself. My goal in this chapter is to perform the standard template filling task, but to first automatically induce the templates (schemas) from an unlabeled corpus.

As discussed previously in this dissertation, there are many ways to represent events, ranging from role-based representations such as frames (Baker et al., 1998) to sequential events in scripts (Schank and Abelson, 1977) and my own narrative
schemas (Chambers and Jurafsky, 2009; Kasch and Oates, 2010). This chapter will learn narrative-like knowledge as schemas, but mapped to the form of IE templates; I learn sets of related events and semantic roles, as shown in this sample output from my system:

**Bombing Template**

\{detonate, blow up, plant, explode, defuse, destroy\}

*Perpetrator:* Person who detonates, plants, blows up  
*Instrument:* Object that is planted, detonated, defused  
*Target:* Object that is destroyed, is blown up

A semantic role, such as *target*, is still a protagonist and a cluster of syntactic functions of the template’s event words (e.g., the objects of *detonate* and *explode*). The algorithm, however, is a slightly modified version of the general schema learning algorithm. I learn templates by first clustering event words based on their proximity in a training corpus. I then perform role induction by clustering the syntactic functions of these events based on selectional preferences and coreferring arguments (i.e., the protagonist). The induced roles are template-specific (e.g., perpetrator), not universal (e.g., agent or patient) or verb-specific. The end representation is a set of related events and a mapping of syntactic arguments to semantic roles.

One of the main contributions of this chapter is that after learning a domain’s template schemas, I perform the standard IE task of role filling template instances from individual documents, for example:

**Perpetrator:** guerrillas  
**Target:** vehicle  
**Instrument:** dynamite

This extraction stage identifies entities using the learned syntactic functions of my roles. I evaluate on the MUC-4 terrorism corpus with results approaching those of supervised systems.
The core of this chapter focuses on how to characterize a domain-specific corpus by learning rich template structure. This differs from the general schema learning in Chapter 3 in that I start from a smaller, more restricted corpus. I describe how to first expand the small corpus’ size, how to cluster its events, and finally how to induce semantic roles. Section 6.5 then describes the extraction algorithm, followed by evaluations against previous supervised work in section 6.6 and 6.7.

6.1 Previous Work

Unsupervised and semi-supervised learning of binary relations and atomic facts has recently received attention. Several approaches learn relations like Person is married to Person without labeled data (Banko et al., 2007b), or rely on a few seed examples for ontology induction (dog is a mammal) and attribute extraction (dogs have tails) (Carlson et al., 2010b,a; Huang and Riloff, 2010; Durme and Pasca, 2008a). However, these approaches focus on atomic relations and not richer template structure.

Algorithms that do focus on template extraction typically require full knowledge of the template structure and labeled corpora, such as rule-based approaches (Chinchor et al., 1993; Rau et al., 1992) and modern supervised classifiers (Freitag, 1998; Chieu et al., 2003; Bunescu and Mooney, 2004; Patwardhan and Riloff, 2009). Classifiers rely on the labeled examples’ surrounding context for features such as nearby tokens, document position, syntax, named entities, semantic classes, and discourse relations (Maslennikov and Chua, 2007). Ji and Grishman (2008) also supplemented labeled data with unlabeled data.

Weakly supervised approaches remove some of the need for fully labeled data. Most still require the templates and their slots. One common approach is to begin with unlabeled, but clustered event-specific documents, and extract common word patterns as extractors (Riloff and Schmelzenbach, 1998; Sudo et al., 2003; Riloff et al., 2005; Patwardhan and Riloff, 2007). Filatova et al. (2006) integrate named entities into pattern learning (PERSON won) to approximate unknown semantic roles. Bootstrapping with seed examples of known slot fillers has been shown to be effective (Surdeanu et al., 2006; Yangarber et al., 2000). In contrast, this chapter removes these
data assumptions, learning instead from a corpus of unknown events and unclustered documents, without seed examples.

Shinyama and Sekine (2006) describe an approach to template learning without labeled data. They present ‘unrestricted relation discovery’ as a means of discovering relations in unlabeled documents, and extract their fillers. Central to the algorithm is collecting multiple documents describing the same exact event (e.g., Hurricane Ivan), and observing repeated word patterns across documents connecting the same proper nouns. Learned patterns represent binary relations, and they show how to construct tables of extracted entities for these relations. My approach draws on this idea of using unlabeled documents to discover relations in text, and of defining semantic roles by sets of entities. However, the limitations to their approach are that (1) redundant documents about specific events are required, (2) relations are binary, and (3) only slots with named entities are learned. I extend their work by showing how to learn without these assumptions, obviating the need for redundant documents, and learning templates with any type and any number of slots.

This chapter is obviously related to my approaches for large-scale learning of scripts and narrative schemas from unlabeled text in Chapters 2 and 3. While I learn interesting event structure from open-domain newspaper articles, the structures typically capture the more frequent topics in a large corpus. I heavily use ideas from previous chapters here, but my goal now is to characterize a specific domain with limited data. Further, this is the first algorithm to apply schema knowledge to the IE task of filling in template mentions in documents.

In summary, this dissertation extends previous work on unsupervised template-based information extraction in a number of ways. The learning algorithm is the first to induce the template structure of a commonly used extraction corpus (the Message Understanding Conference corpus), and it is the first to extract entities without knowing how many templates exist, without examples of slot fillers, and without event-clustered documents.
6.2 The MUC-4 Corpus

My goal is to learn the general event structure of a domain, and then extract the instances of each learned event. In order to measure performance in both tasks (learning structure and extracting instances), I use the terrorism corpus of the Message Understanding Conference (MUC-4) as my target domain (Sundheim, 1991). This corpus of newspaper articles from Latin America was chosen because the documents are annotated with templates that describe all of the entities involved in each event. An example snippet from a Bombing document is given here:

LIMA, 11 Mar 89 ( EFE ) – ( text ) it was officially reported today that on 10 March alleged Shining Path members murdered four members of a family in Santa Maria del Valle, Huanuco department. After killing the peasant family, the terrorists used explosives against the town hall. El Comercio reported that alleged Shining Path members also attacked public facilities in huarpacha, Ambo, tomayquichua, and kichki. Municipal official Sergio Horna was seriously wounded in an explosion in Ambo.

The main entities in this document fill the slots of a MUC-4 Bombing template, as shown in Figure 6.2. The document’s main entities (Shining Path members and Sergio Horna) fill string-based slots (perpetrator (PERP) and human target (HUM TGT)). Other properties about this document’s events, such as the date and location, similarly fill their respective slots. Finally, the ‘INCIDENT: TYPE’ slot indicates the type of template (e.g., Bombing).

The MUC-4 corpus defines six template types: Attack, Kidnapping, Bombing, Arson, Robbery, and Forced Work Stoppage. A document can be labeled with more than one template and type. This short example document is actually labeled with two Bombing templates and three Attack templates. One bombing template includes the bombing of a town hall and public facilities in Santa Maria Del Valle (second sentence), and the other is a bombing of public facilities in Huarpacha (third sentence). The first Attack template covers the first sentence’s attack on ‘members of a family’. The remaining three Attack templates cover the third sentence, duplicating
<table>
<thead>
<tr>
<th></th>
<th>Event Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>MESSAGE: ID</td>
<td>DEV-MUC3-0112 (BELLCORE, MITRE)</td>
</tr>
<tr>
<td>1.</td>
<td>MESSAGE: TEMPLATE</td>
<td>4</td>
</tr>
<tr>
<td>2.</td>
<td>INCIDENT: DATE</td>
<td>10 MAR 89</td>
</tr>
<tr>
<td>3.</td>
<td>INCIDENT: LOCATION</td>
<td>PERU: HUANUCO (DEPARTMENT): AMBO (TOWN)</td>
</tr>
<tr>
<td>4.</td>
<td>INCIDENT: TYPE</td>
<td>BOMBING</td>
</tr>
<tr>
<td>5.</td>
<td>INCIDENT: STAGE OF EXECUTION</td>
<td>ACCOMPLISHED</td>
</tr>
<tr>
<td>6.</td>
<td>INCIDENT: INSTRUMENT ID</td>
<td>-</td>
</tr>
<tr>
<td>7.</td>
<td>INCIDENT: INSTRUMENT TYPE</td>
<td>EXPLOSIVE: &quot;-&quot;</td>
</tr>
<tr>
<td>8.</td>
<td>PERP: INCIDENT CATEGORY</td>
<td>TERRORIST ACT</td>
</tr>
<tr>
<td>9.</td>
<td>PERP: INDIVIDUAL ID</td>
<td>&quot;SHINING PATH MEMBERS&quot;</td>
</tr>
<tr>
<td>10.</td>
<td>PERP: ORGANIZATION ID</td>
<td>&quot;SHINING PATH&quot;</td>
</tr>
<tr>
<td>11.</td>
<td>PERP: ORGANIZATION CONFIDENCE</td>
<td>SUSPECTED OR ACCUSED: &quot;SHINING PATH&quot;</td>
</tr>
<tr>
<td>12.</td>
<td>PHYS TGT: ID</td>
<td>&quot;PUBLIC FACILITIES&quot;</td>
</tr>
<tr>
<td>13.</td>
<td>PHYS TGT: TYPE</td>
<td>OTHER: &quot;PUBLIC FACILITIES&quot;</td>
</tr>
<tr>
<td>14.</td>
<td>PHYS TGT: NUMBER</td>
<td>PLURAL: &quot;PUBLIC FACILITIES&quot;</td>
</tr>
<tr>
<td>15.</td>
<td>PHYS TGT: FOREIGN NATION</td>
<td>-</td>
</tr>
<tr>
<td>16.</td>
<td>PHYS TGT: EFFECT OF INCIDENT</td>
<td>-</td>
</tr>
<tr>
<td>17.</td>
<td>PHYS TGT: TOTAL NUMBER</td>
<td>-</td>
</tr>
<tr>
<td>18.</td>
<td>HUM TGT: NAME</td>
<td>&quot;SERGIO HORNA&quot;</td>
</tr>
<tr>
<td>19.</td>
<td>HUM TGT: DESCRIPTION</td>
<td>&quot;MUNICIPAL OFFICIAL&quot;: &quot;SERGIO HORNA&quot;</td>
</tr>
<tr>
<td>20.</td>
<td>HUM TGT: TYPE</td>
<td>GOVERNMENT OFFICIAL: &quot;SERGIO HORNA&quot;</td>
</tr>
<tr>
<td>21.</td>
<td>HUM TGT: NUMBER</td>
<td>1: &quot;SERGIO HORNA&quot;</td>
</tr>
<tr>
<td>22.</td>
<td>HUM TGT: FOREIGN NATION</td>
<td>-</td>
</tr>
<tr>
<td>23.</td>
<td>HUM TGT: EFFECT OF INCIDENT</td>
<td>INJURY: &quot;SERGIO HORNA&quot;</td>
</tr>
<tr>
<td>24.</td>
<td>HUM TGT: TOTAL NUMBER</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 6.1: A template from the MUC-4 corpus.
QUITO, 8 Mar 89 (EFE) — The Ecuadoran insurgent group “free homeland Montoneros” (MPL) today reiterated that it will not abandon its armed struggle. It made this announcement 1 day after the “Alfaro lives, damnit” (AVC) group signed a peace agreement with President Rodrigo Borja’s government. The MPL today expressed its determination to continue its armed struggle and not to follow the AVC’s footsteps. AVC leaders and two Ecuadoran ministers yesterday appeared before the media to formalize the peace agreement between the AVC and the government. In a communique released to EFE, The MPL states that it will now brandish its weapons “with greater force to attain freedom for Ecuadoran society and its people.”

Figure 6.2: A portion of a MUC-4 document with no templates to extract.

The MUC-4 training corpus contains 1300 documents. 733 of the documents are labeled with at least one template. While there are six types of templates, only four are modestly frequent: bombing (208 docs), kidnap (83 docs), attack (479 docs), and arson (40 docs). 567 documents are not labeled with any templates. These latter unlabeled documents are miscellaneous articles that report on non-specific political events and speeches. Figure 6.2 gives an example of an unlabeled document.

6.3 A Narrative Approach to Extraction

The template structure in MUC-4 is conducive to narrative learning because it defines the central entities and their semantic roles in each situation. Narrative schemas learn similar semantic roles, but with the added advantage of learning which event words trigger those roles in each situation (e.g., a Bombing includes detonate, explode,
plant, etc.). In order to evaluate how narrative schemas can assist this Information Extraction application, I will compare my induced structure to the same template structures used by state-of-the-art MUC-4 systems.

Although the previous section gave an example of a full MUC-4 template, most current systems do not evaluate performance on extracting values for the entire template. Current extraction evaluations instead focus solely on the four string-based slots in the templates, ignoring the other parameterized slots that involve deeper reasoning (such as ‘stage of execution’ and ‘effect of incident’). The four main slots from Figure 6.2 and examples of entity fillers are shown here:

- **Perpetrator**: Shining Path members
- **Victim**: Sergio Horna
- **Target**: public facilities
- **Instrument**: explosives

Current extraction algorithms assume they know these four slot types in advance (e.g., perpetrator, victim, target, instrument), and focus solely on learning to extract their fillers from documents using supervised techniques. I will also extract the fillers, but I do not start with knowledge of the four slots, but instead induce this structure automatically.

I induce this template structure from the MUC-4 corpus, and compare my induced structure to these same four slots\(^1\) As described above, the training corpus consists of 1300 documents, but 567 of them are not labeled with any templates. My learning algorithm does not know which documents contain (or do not contain) which templates. After learning event words that represent templates, I induce their slots, not knowing a priori how many there are, and then fill them in by extracting entities as in the standard task. In the document example above, the algorithm will learn to identify three verbs (use, attack, wound) that indicate the Bombing template, and their syntactic arguments fill its slots.

\(^1\)There are two Perpetrator slots in MUC-4: Organization and Individual. I consider their union as a single slot.
CHAPTER 6. LEARNING EVENTS FOR INFORMATION EXTRACTION

6.4 Learning Templates from Raw Text

My goal is to learn templates that characterize a domain as described in unclustered, unlabeled documents. This presents a two-fold problem to the learner: it does not know how many events exist, and it does not know which documents describe which event (some may describe multiple events). I approach this problem with a three step process: (1) cluster the domain’s event patterns to approximate the template topics, (2) build a new corpus specific to each cluster by retrieving documents from a larger unrelated corpus, (3) induce each template’s slots using its new (larger) corpus of documents.

6.4.1 Clustering Events to Learn Templates

I cluster event patterns to create templates. An event pattern is either (1) a verb, (2) a noun in WordNet under the Event synset, or (3) a verb and the head word of its syntactic object. Examples of each include (1) ‘explode’, (2) ‘explosion’, and (3) ‘explode:bomb’. I also tag the corpus with the Stanford NER system (Finkel et al., 2005) and allow patterns to include named entity types, e.g., ‘kidnap:PERSON’. These patterns are crucially needed later to learn a template’s slots. However, I first need an algorithm to cluster these patterns to learn the domain’s core events. I consider two unsupervised algorithms: Latent Dirichlet Allocation (Blei et al., 2003) and agglomerative clustering based on word distance.

LDA for Unknown Data

Latent Dirichlet Allocation (LDA) is a probabilistic model that treats documents as mixtures of topics. It learns topics as discrete distributions (multinomials) over the event patterns, and thus meets my needs as it clusters patterns based on co-occurrence in documents. The algorithm requires the number of topics to be known ahead of time, but in practice this number is set relatively high and the resulting topics are still useful. My best performing LDA model used 200 topics. I had mixed success with LDA though, and ultimately found my next approach performed slightly better on the document classification evaluation.
Clustering on Event Distance

Agglomerative clustering does not require foreknowledge of the templates, but its success relies on how event pattern similarity is determined.

Ideally, I want to learn that `detonate` and `destroy` belong in the same cluster representing a bombing. Vector-based approaches are often adopted to represent words as feature vectors and compute their distance with cosine similarity. Unfortunately, these approaches typically learn clusters of synonymous words that can miss detonate and destroy. My goal is to instead capture world knowledge of co-occurring events. I thus adopt an assumption that `closeness` in the world is reflected by `closeness` in a text’s discourse. I hypothesize that two patterns are related if they occur near each other in a document more often than chance.

Let \( g(w_i, w_j) \) be the distance between two events (1 if in the same sentence, 2 in neighboring, etc). Let \( C_{\text{dist}}(w_i, w_j) \) be the distance-weighted frequency of two events occurring together:

\[
C_{\text{dist}}(w_i, w_j) = \sum_d \sum_{w_i, w_j \in D} 1 - \log_4(g(w_i, w_j)) \tag{6.1}
\]

where \( d \) is a document in the set of all documents \( D \). The base 4 logarithm discounts neighboring sentences by 0.5 and scores those within the same sentence as 1. Using this definition of distance, pointwise mutual information measures the similarity of any two events:

\[
\text{pmi}(w_i, w_j) = \frac{P_{\text{dist}}(w_i, w_j)}{P(w_i)P(w_j)} \tag{6.2}
\]

\[
P(w_i) = \frac{C(w_i)}{\sum_j C(w_j)} \tag{6.3}
\]

\[
P_{\text{dist}}(w_i, w_j) = \frac{C_{\text{dist}}(w_i, w_j)}{\sum_k \sum_l C_{\text{dist}}(w_k, w_l)} \tag{6.4}
\]

I run agglomerative clustering with \( \text{pmi} \) over all event patterns. Merging decisions use the average link score between all new links across two clusters. As with all clustering algorithms, a stopping criterion is needed. I continue merging clusters until any single cluster grows beyond \( m \) patterns. I briefly inspected the clustering process
and chose \( m = 40 \) to prevent learned scenarios from intuitively growing too large and ambiguous. Post-evaluation analysis shows that this value has wide flexibility. For example, the Kidnap and Arson clusters are unchanged from \( 30 < m < 80 \), and Bombing unchanged from \( 30 < m < 50 \). Figure 6.3 shows 3 clusters (of 77 learned) that characterize the main template types.

### 6.4.2 Information Retrieval for Templates

Learning a domain often suffers from a lack of training data. The previous section clustered events from the MUC-4 corpus, but its 1300 documents do not provide enough examples of verbs and argument counts to further learn the semantic roles in each cluster. My solution is to assemble a larger IR-corpus of documents for each cluster. For example, MUC-4 labels 83 documents with Kidnap, but my learned cluster (\( \text{kidnap}, \text{abduct}, \text{release}, ... \)) retrieved 3954 documents from a general corpus.

I use the Associated Press and New York Times sections of the Gigaword Corpus (Graff, 2002) as my general corpus. These sections include approximately 3.5 million news articles spanning 12 years.

My retrieval algorithm retrieves documents that score highly with a cluster’s tokens. A document’s score is defined by two common metrics: word match, and word coverage. A document’s match score is defined as the average number of times the words in cluster \( c \) appear in document \( d \):

\[
\text{match}(d, c) = \frac{\sum_{w \in c} \sum_{t \in d} 1\{w = t\}}{|c|} \tag{6.5}
\]
I define coverage as the number of seen cluster words. Coverage avoids matching documents that score highly by repeating a single cluster word a lot. I only score a document if its coverage, $\text{coverage}(d, c)$, is at least 3 words (or less for tiny clusters):

$$
ir(d, c) = \begin{cases} 
\text{match}(d, c) & \text{if } \text{coverage}(d, c) > \min(3, \frac{|c|}{4}) \\
0 & \text{otherwise}
\end{cases}
$$

A document $d$ is retrieved for a cluster $c$ if $ir(d, c) > 0.4$. Finally, I emphasize precision by pruning away 50% of a cluster’s retrieved documents that are farthest in distance from the mean document of the retrieved set. Distance is just the cosine similarity between bag-of-words vector representations. The confidence value of 0.4 was chosen from a brief manual inspection among a single cluster’s retrieved documents. Pruning 50% was arbitrarily chosen to improve precision, and I did not experiment with other pruning amounts. A search for optimum values among these parameters may possibly lead to better results.

### 6.4.3 Inducing Semantic Roles (Slots)

Having successfully clustered event words and retrieved an IR-corpus for each cluster, I now address the problem of inducing semantic roles. My learned roles will then extract entities in the next section and I will evaluate their per-role accuracy.

Most work on unsupervised role induction focuses on learning verb-specific roles, starting with seed examples (Swier and Stevenson, 2004b; He and Gildea, 2006b) and/or knowing the number of roles (Grenager and Manning, 2006b; Lang and Lapata, 2010). This dissertation has shown how to learn situation-specific roles over scenarios (Chambers and Jurafsky, 2009), similar to frame roles in FrameNet (Baker et al., 1998). I link the syntactic relations of verbs by clustering them based on observing coreferring arguments in those positions. This chapter is now extending this intuition by introducing a new vector-based approach to coreference similarity.
Syntactic Relations as Roles

I learn the roles of cluster $C$ by clustering the syntactic relations $R_C$ of its words. Consider the following example:

$$C = \{\text{go off, explode, set off, damage, destroy}\}$$
$$R_C = \{\text{go\_off:s, go\_off:p\_in, explode:s, set\_off:s}\}$$

where $\text{verb:s}$ is the verb’s subject, $\text{:o}$ the object, and $\text{:p\_in}$ a preposition. I ideally want to cluster $R_C$ as:

$$\text{bomb} = \{\text{go\_off:s, explode:s, set\_off:o, destroy:s}\}$$
$$\text{suspect} = \{\text{set\_off:s}\}$$
$$\text{target} = \{\text{go\_off:p\_in, destroy:o}\}$$

I want to cluster all subjects, objects, and prepositions. Passive voice is normalized to active\(^2\).

I adopt two views of relation similarity: coreferring arguments and selectional preferences. As I have observed earlier concerning the protagonist, coreferring arguments suggest a semantic relation between two predicates. In the sentence, *he ran and then he fell*, the subjects of run and fall corefer, and so they likely belong to the same scenario-specific semantic role. I applied this idea to a new vector similarity framework. I represent a relation as a vector of all relations with which their arguments coreferred. For instance, arguments of the relation $\text{go\_off:s}$ were seen coreferring with mentions in $\text{plant:o, set\_off:o}$ and $\text{injure:s}$. I represent $\text{go\_off:s}$ as a vector of these relation counts, calling this its *coref vector representation*. Table 6.1 shows two more examples: coreference vectors for the two subjects of kidnap and release. When compared using cosine similarity, these two are scored highly (0.93).

Selectional preferences (SPs) are also useful in measuring similarity (Erk and Pado, 2008). A verb and its syntactic function can be represented as a vector of the observed arguments filling that function during training. For instance, the SPs for $\text{go\_off:s}$ in my data include $\{\text{bomb, device, charge, explosion}\}$. An SP vector is thus a vector of argument head words and their corpus counts, multiplied by the argument’s

---

\(^2\)I use the ‘Stanford Parser’ at nlp.stanford.edu/software
CHAPTER 6. LEARNING EVENTS FOR INFORMATION EXTRACTION

Coref Vector Representation

<table>
<thead>
<tr>
<th>kidnap:subject</th>
<th>release:subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>kill:subject</td>
<td>hold:subject</td>
</tr>
<tr>
<td>hold:subject</td>
<td>demand:subject</td>
</tr>
<tr>
<td>release:subject</td>
<td>kill:subject</td>
</tr>
<tr>
<td>threaten:subject</td>
<td>tell:subject</td>
</tr>
<tr>
<td>rule:prep_by</td>
<td>seize:subject</td>
</tr>
<tr>
<td>try:subject</td>
<td>threaten:subject</td>
</tr>
<tr>
<td>seize:subject</td>
<td>hostage:possessive</td>
</tr>
<tr>
<td>negotiate:subject</td>
<td>release:possessive</td>
</tr>
</tbody>
</table>

Table 6.1: Coreference Vectors for the subjects of kidnap and release. The syntactic functions are counts of how many times each syntactic function contained an argument that coreferred with the main verb’s subject (e.g., kidnap or release).

inverse document frequency (IDF) to penalize common words. Using the same two verbs as examples, *kidnap* and *release*, Table 6.2 shows the top preferences for each. I show only the corpus counts for readability, leaving out the IDF factor.

*Kidnap* and *release* have similar selectional preference vectors, but the similarity is not as strong as their coreference vectors. In addition to the normal kidnapper words, the subject of *release* includes many countries (the LOCATION named entity tag) and governments. This is because countries and governments often *release* records and reports, but have little to do with kidnapping events. The cosine similarity score in this case is thus penalized from the non-kidnap events that *release* often associates with.

I measure similarity using cosine similarity in both approaches, but coreference vectors and selectional preference vectors are measuring different types of similarity. Coreference is a looser narrative similarity (bombings cause injuries), while SPs capture synonymy (plant and place have similar arguments). I observed that many narrative relations are not synonymous, and vice versa. I thus take the maximum of either cosine score as our final similarity metric between two relations. I then back off to the average of the two cosine scores if the max is not confident (less than 0.7). When neither score is confident, taking the average further penalizes the similarity score. I chose the value of 0.7 from a grid search to optimize extraction results on
CHAPTER 6. LEARNING EVENTS FOR INFORMATION EXTRACTION

Selectional Preferences Vector

<table>
<thead>
<tr>
<th>kidnap:subject</th>
<th>release:subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>rebel</td>
<td>283</td>
</tr>
<tr>
<td>group</td>
<td>246</td>
</tr>
<tr>
<td>ORGANIZATION</td>
<td>242</td>
</tr>
<tr>
<td>gunman</td>
<td>199</td>
</tr>
<tr>
<td>militant</td>
<td>151</td>
</tr>
<tr>
<td>tribesman</td>
<td>149</td>
</tr>
<tr>
<td>insurgent</td>
<td>140</td>
</tr>
<tr>
<td>man</td>
<td>133</td>
</tr>
<tr>
<td>kidnapper</td>
<td>223</td>
</tr>
<tr>
<td>LOCATION</td>
<td>162</td>
</tr>
<tr>
<td>rebel</td>
<td>151</td>
</tr>
<tr>
<td>group</td>
<td>123</td>
</tr>
<tr>
<td>PERSON</td>
<td>122</td>
</tr>
<tr>
<td>ORGANIZATION</td>
<td>119</td>
</tr>
<tr>
<td>government</td>
<td>77</td>
</tr>
<tr>
<td>militant</td>
<td>54</td>
</tr>
</tbody>
</table>

Table 6.2: Selectional Preference vectors for the subjects of kidnap and release.

the training set. In the kidnap/release example above, the coreference vector cosine score (0.93) is selected as their final similarity score.

Clustering Syntactic Functions

I use agglomerative clustering with the above pairwise similarity metric. Cluster similarity is the average link score over all new links crossing two clusters. I include a sparsity penalty \( r(c_a, c_b) \) if there are too few links between clusters \( c_a \) and \( c_b \): Clustering stops when the merged cluster scores drop below a threshold optimized to extraction performance on the training data.

\[
score(c_a, c_b) = \sum_{w_i \in c_a} \sum_{w_j \in c_b} \text{sim}(w_i, w_j) \ast r(c_a, c_b) \\
\]

\[
r(c_a, c_b) = \frac{\sum_{w_i \in c_a} \sum_{w_j \in c_b} 1\{\text{sim}(w_i, w_j) > 0\}}{\sum_{w_i \in c_a} \sum_{w_j \in c_b} 1} \\
\]

This penalizes clusters from merging when they share only a few high scoring edges.

I also begin with two assumptions about syntactic functions and semantic roles. The first assumes that the grammatical subject and object of a verb carry different semantic roles. For instance, the subject of \textit{sell} fills a different role (Seller) than the object (Good). The second assumption is that each semantic role has a high-level entity type. For instance, the subject (or agent role) of \textit{sell} is a Person or
Organization, and the object is a Physical Object.

I implement the first assumption as a constraint in the clustering algorithm, preventing two clusters from merging if their union contains the same verb’s subject and object.

I implement the second assumption by automatically labeling each syntactic function with a role type based on its observed arguments. The role types are broad general classes: Person/Org, Physical Object, or Other. A syntactic function is labeled as a class if 20% of its arguments appear under the corresponding WordNet synset\(^3\), or if the NER system labels them as such. Once labeled by type, I separately cluster the syntactic functions for each role type. For instance, Person functions are clustered separate from Physical Object functions. Figures 6.4 and 6.5 show some of the resulting roles.

Finally, since agglomerative clustering makes hard decisions, related events to a template may have been excluded in the initial event clustering stage. To address this problem, I identify the 200 nearby events to each event cluster. These are simply the top scoring event patterns with the cluster’s original events. I add their syntactic functions to their best matching roles. This expands the coverage of each learned role. Varying the 200 amount does not lead to wide variation in extraction performance. Once induced, the roles are evaluated by their entity extraction performance in Section 6.5.

6.4.4 Template Evaluation

I now compare my learned templates to those hand-created by human annotators for the MUC-4 terrorism corpus. The corpus contains 6 template types, but two of them occur in only 4 and 14 of the 1300 training documents. I thus only evaluate the 4 main templates (bombing, kidnapping, attack, and arson), ignoring robbery and forced work stoppage. The gold slots are shown in figure 6.6.

I evaluate the four learned templates that score highest in the document classification evaluation (to be described in section 6.5.1), aligned with their MUC-4 types.

\(^3\)Physical objects are defined as non-person physical objects
CHAPTER 6. LEARNING EVENTS FOR INFORMATION EXTRACTION

**Bombing Template** (MUC-4)

**Perpetrator** Person/Org who detonates, blows up, plants, hurls, stages, is detained, is suspected, is blamed on, launches

**Instrument** A *physical object* that is exploded, explodes, is hurled, causes, goes off, is planted, damages, is set off, is defused

**Target** A *physical object* that is damaged, is destroyed, is exploded at, is damaged, is thrown at, is hit, is struck

**Police** Person/Org who raids, questions, discovers, investigates, defuses, arrests

**N/A** A *physical object* that is blown up, destroys

**Attack/Shooting Template** (MUC-4)

**Perpetrator** Person/Org who assassinates, patrols, ambushed, raids, shoots, is linked to

**Victim** Person/Org who is assassinated, is toppled, is gunned down, is executed, is evacuated

**Target** Person/Org who is hit, is struck, is downed, is set fire to, is blown up, surrounded

**Instrument** A *physical object* that is fired, injures, downs, is set off, is exploded

**Kidnap Template** (MUC-4)

**Perpetrator** Person/Org who releases, abducts, kidnaps, ambushes, holds, forces, captures, is imprisoned, frees

**Target** Person/Org who is kidnapped, is released, is freed, escapes, disappears, travels, is harmed, is threatened

**Police** Person/Org who rules out, negotiates, condemns, is pressured, finds, arrests, combs

Figure 6.4: Three learned example templates representing three of the original MUC-3 hand-coded templates.
### Weapons Smuggling Template (NEW)

**Perpetrator** *Person/Org* who smuggles, is seized from, is captured, is detained

**Police** *Person/Org* who raids, seizes, captures, confiscates, detains, investigates

**Instrument** A *physical object* that is smuggled, is seized, is confiscated, is transported

### Election Template (NEW)

**Voter** *Person/Org* who chooses, is intimidated, favors, is appealed to, turns out

**Government** *Person/Org* who authorizes, is chosen, blames, authorizes, denies

**Candidate** *Person/Org* who resigns, unites, advocates, manipulates, pledges, is blamed

Figure 6.5: Two learned example templates that were not hand-created in the MUC-3 corpus. All knowledge except the template/role names (e.g., ‘Victim’) is learned.

<table>
<thead>
<tr>
<th></th>
<th>Bombing</th>
<th>Kidnap</th>
<th>Attack</th>
<th>Arson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perpetrator</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Victim</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Target</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Instrument</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Figure 6.6: Slots in the hand-crafted MUC-4 templates.
Figure 6.4 shows three of my four templates, and figure 6.5 two brand new ones that my algorithm learned. Of the four templates, I learned 12 of the 13 semantic roles as created for MUC. In addition, I learned a new role not in MUC-4 for bombings, kidnappings, and arson: the Police or Authorities role. The annotators chose not to include this in their labeling, but this knowledge is clearly relevant when understanding such events, so I consider it correct. There is one additional Bombing and one Arson role that does not align with MUC-4, marked incorrect. I thus report 92% slot recall, and precision as 14 of 16 (88%) learned slots.

I only measure agreement with the MUC-4 template schemas here, but my system learns other events as well. I show two such examples in figure 2: the Weapons Smuggling and Election Templates. This illustrates the strength in using a learning approach based on narrative schemas. I have learned brand new knowledge (elections and smuggling) that event human annotators did not capture when they originally annotated the corpus. Human annotations are expensive, and typically only the main events are labeled. A fully automated knowledge extraction algorithm such as this one can greatly expand the knowledge we extract.

6.5 Information Extraction: Slot Filling

I now present how to apply my learned templates to information extraction. This section will describe how to extract slot fillers using my templates, but without knowing which templates are correct.

I could simply use a standard IE approach, for example, creating seed words for my new learned templates. But instead, I propose a new method that obviates the need for even a limited human labeling of seed sets. I consider each learned semantic role as a potential slot, and I extract slot fillers using the syntactic functions that were previously learned. Thus, the learned syntactic patterns (e.g., the subject of release) serve the dual purpose of both inducing the template slots, and extracting appropriate slot fillers from text.
6.5.1 Document Classification

A document is labeled for a template if two different conditions are met: (1) it contains at least one trigger phrase, and (2) its average per-token conditional probability meets a strict threshold.

Both conditions require a definition of the probability of a token and a template. The probability is then defined as the token’s importance relative to its uniqueness across all templates:

\[
P(w, t) = \frac{P_{IR_t}(w)}{\sum_{s \in T} P_{IR_s}(w)}
\]

where \(P_{IR_t}(w)\) is the probability of pattern \(w\) in the IR-corpus of template \(t\).

\[
P_{IR_t}(w) = \frac{C_t(w)}{\sum_v C_t(v)}
\]

where \(C_t(w)\) is the number of times word \(w\) appeared in the IR-corpus of template \(t\). A template’s trigger words are then defined as all words satisfying \(P(w, t) > 0.2\). Trigger phrases are thus template-specific patterns that are highly indicative of that template.

After identifying triggers, I use the above definition to score a document with a template. A document is labeled with a template if it contains at least one trigger, and its average word probability is greater than a parameter optimized on the training set. A document can be (and often is) labeled with multiple templates.

Finally, I label the sentences that contain triggers and use them for extraction in section 6.5.2.

Experiment: Document Classification

The MUC-4 corpus links templates to documents, allowing us to evaluate my document labels. I treat each link as a gold label (kidnap, bomb, attack, or arson) for that document, and documents can have multiple labels. My learned clusters naturally do not have MUC-4 labels, so I report results on the four clusters that score highest with each label. I evaluate each cluster’s document assignments with each gold label, and map the cluster with the highest performance to each label.
### Document Classification

<table>
<thead>
<tr>
<th></th>
<th>Kidnap</th>
<th>Bomb</th>
<th>Attack</th>
<th>Arson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>.64</td>
<td>.83</td>
<td>.66</td>
<td>.30</td>
</tr>
<tr>
<td>Recall</td>
<td>.54</td>
<td>.63</td>
<td>.35</td>
<td>1.0</td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td><strong>.58</strong></td>
<td><strong>.72</strong></td>
<td><strong>.46</strong></td>
<td><strong>.46</strong></td>
</tr>
</tbody>
</table>

Table 6.3: Document classification results on test.

Table 6.3 shows the document classification results for precision, recall, and F1 score. The bombing template performs best with an F1 score of .72, and kidnap with .58. Precision is above .60 for the three most prevalent templates.

The classification results for the Attack template is lower because the MUC-4 definition of an attack is essentially an agglomeration of diverse types of attack events. For instance, documents that describe single murders, broader military coups, and infrastructure sabotage are all labeled with the same general Attack template. My learned templates, however, have a different granularity. Rather than one broad Attack type, I learn several: Shooting, Murder, Coup, General Injury, and Pipeline Attack. I see these subtypes as strengths of my algorithm, but it misses the MUC-4 granularity of Attack and results in a lower F1 score.

### 6.5.2 Entity Extraction

Once documents are labeled with templates, I next extract entities into the template slots. Extraction occurs in the trigger sentences from the previous section. The extraction process is two-fold:

1. Extract all NPs that are arguments of patterns in the template’s induced roles.

2. Extract NPs whose heads are observed frequently with one of the roles (e.g., ‘bomb’ is seen with Instrument relations in figure 6.4).

Take the following MUC-4 sentence as an example:

*The two bombs were planted with the exclusive purpose of intimidating the owners of the premises.*
The verb *plant* is in my learned bombing cluster, so step (1) will extract its passive subject *bombs* and map it to the correct instrument role (see figure 6.4). The human target, *owners*, is missed because *intimidate* was not learned. However, if *owner* is in the selectional preferences of the learned ‘human target’ role, step (2) correctly extracts it into that role.

These are two different, but complementary, views of semantic roles. The first is that a role is defined by the set of syntactic relations that describe it. Thus, I find all role relations and save their arguments (pattern extraction). The second view is that a role is defined by the arguments that fill it. Thus, I extract all arguments that filled a role in training, regardless of their current syntactic environment.

Finally, I filter extractions whose WordNet or named entity label does not match the learned slot’s type (e.g., a Location does not match a Person).

### 6.6 Standard Evaluation

I trained on the 1300 documents in the MUC-4 corpus and tested on the 200 document TST3 and TST4 test set. I evaluate the four string-based slots: perpetrator, physical target, human target, and instrument. I merge MUC’s two perpetrator slots (individuals and orgs) into one gold Perpetrator slot. As in Patwardhan and Riloff (2007; 2009), I ignore missed optional slots in computing recall. I induced clusters in training, performed IR, and induced the slots. I then extracted entities from the test documents following the above procedure in section 6.5.2.

The standard evaluation for this corpus is to report the F1 score for slot type accuracy, ignoring the template type. For instance, a perpetrator of a bombing and a perpetrator of an attack are treated the same. This allows supervised classifiers to train on all perpetrators at once, rather than template-specific learners. Although not ideal for my learning goals, I report it for comparison against previous work.

Several supervised approaches have presented results on MUC-4, but unfortunately I cannot compare against them. Maslennikov and Chua (2006; 2007) evaluated a random subset of test (they report .60 and .63 F1), and Xiao et al. (2004) did not evaluate all slot types (they report .57 F1 on perpetrator, physical target, and human
MUC-4 Extraction Results

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patwardhan &amp; Riloff-09: Supervised</td>
<td>48</td>
<td>59</td>
<td>53</td>
</tr>
<tr>
<td>Patwardhan &amp; Riloff-07: Weak-Sup</td>
<td>42</td>
<td>48</td>
<td>44</td>
</tr>
<tr>
<td>My Results (1 attack)</td>
<td>48</td>
<td>25</td>
<td>33</td>
</tr>
<tr>
<td>My Results (5 attack)</td>
<td>44</td>
<td>36</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 6.4: MUC-4 extraction, ignoring template type.

target, but did not include instrument).

Table 6.4 thus shows my results with previous work that is comparable: the fully supervised and weakly supervised approaches of Patwardhan and Riloff (2009; 2007). I give two numbers for my system: mapping one learned template to Attack, and mapping five. As discussed above, my learned templates for Attack have a different granularity than MUC-4. My system actually learns several distinct types representing specific attack situations like Shooting, Murder, Coup, General Injury, and Pipeline Attack. I thus show results when I apply the best five learned templates to Attack, rather than just one. The final F1 with these Attack subtypes is .40.

The precision of my algorithm is as good as (and its F1 score near) two supervised algorithms that require knowledge of the templates and/or labeled data. My algorithm instead learned this knowledge without such supervision. The main reason for the slightly lower F1 score from previous work is recall. Supervised approaches can take advantage of knowing that only four templates exist, and so can more easily identify their key sentences. A sentence describes one of the four templates, or none of them. My approach instead learned over 70 potential templates, many of which are noisy. Identifying key sentences is thus more difficult as there are over 70 classes to choose from for each sentence, and recall is hurt as the algorithm skips sentences to maintain similar precision to supervised approaches.
### Individual Template Performance

<table>
<thead>
<tr>
<th>F1 Score</th>
<th>Kidnap</th>
<th>Bomb</th>
<th>Arson</th>
<th>Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results</td>
<td>.53</td>
<td>.43</td>
<td>.42</td>
<td>.16 / .25</td>
</tr>
</tbody>
</table>

Table 6.5: Performance of the individual templates. The Attack column compares both of my 1 vs 5 best templates.

### Individual Template Performance on Gold Documents

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kidnap</td>
<td>.82</td>
<td>.47</td>
<td>.60 (+.07)</td>
</tr>
<tr>
<td>Bomb</td>
<td>.60</td>
<td>.36</td>
<td>.45 (+.02)</td>
</tr>
<tr>
<td>Arson</td>
<td>1.0</td>
<td>.29</td>
<td>.44 (+.02)</td>
</tr>
<tr>
<td>Attack</td>
<td>.36</td>
<td>.09</td>
<td>.15 (0.0)</td>
</tr>
</tbody>
</table>

Table 6.6: Performance of each template type, but only evaluated on the documents that are labeled with that template type. All others are removed from testing. The parentheses indicate F1 score gain over evaluating on all test documents (Table 6.5).

### 6.7 Specific Evaluation

In order to more precisely evaluate each learned template, I also evaluated per-template performance. Instead of merging all slots across all template types, I score the slots within each template type. This is a stricter evaluation than Section 6.6; for example, bombing victims assigned to attacks were previously deemed correct.

Table 6.5 gives my results. Three of the four templates score at or above .42 F1, showing that my lower score from the previous section is mainly due to the Attack template. Arson also unexpectedly scored well. It only occurs in 40 documents overall, suggesting my algorithm works with little evidence.

Per-template performance is good, and my .40 overall score from the previous section illustrates that I perform quite well in comparison to the .44-.53 range of weakly and fully supervised results.

---

4I do not address the task of template instance identification (e.g., splitting two bombings into separate instances). This requires deeper discourse analysis not addressed by this dissertation.
These evaluations use the standard TST3 and TST4 test sets, including the documents that are not labeled with any templates. 74 of the 200 test documents are unlabeled. In order to determine where the system’s false positives originate, I also measure performance only on the 126 test documents that have at least one template. Figure 6.6 presents the results on this subset. Kidnap improves most significantly in F1 score (7 F1 points absolute), but the others only change slightly. Most of the false positives in the system thus do not originate from the unlabeled documents (the 74 unlabeled), but rather from extracting incorrect entities from correctly identified documents (the 126 labeled).

6.8 Discussion

Template-based IE systems typically assume knowledge of the domain and its templates. I began by showing that domain knowledge isn’t necessarily required; I learned the MUC-4 template structure with surprising accuracy, learning new semantic roles and several new template structures. This algorithm is the first to my knowledge to automatically induce MUC-4 templates. It is possible to take these learned slots and use a previous approach to IE (such as seed-based bootstrapping), but I presented an algorithm that instead uses my learned syntactic patterns. I achieved results with comparable precision, and an F1 score of .40 that approaches prior algorithms that rely on hand-crafted knowledge.

Future work will need to address how the algorithm’s unsupervised nature hurts recall. Without labeled or seed examples, it does not learn as many patterns or robust classifiers as supervised approaches. I will investigate new text sources and algorithms to try and capture more knowledge. The final experiment in figure 6.6 shows that perhaps new work should first focus on the pattern learning and entity extraction algorithm, rather than document identification. Given perfect document identification, system recall still suffers, suggesting that bigger gains can be found by improving the extraction algorithm, rather than document identification.
The extraction results are encouraging, but the template induction itself is a central contribution of this work. My induction algorithm is the first to learn the template knowledge structures for the MUC-4 task, and my algorithm learned additional template types (e.g., government elections) not annotated by the MUC-4 annotators. A user seeking information in a body of text would recognize these new templates, and could then extract the central entities. Knowledge induction plays an important role in moving to new domains and assisting users who may not know what a corpus contains. Recently, there is significant activity in the area of unsupervised knowledge induction, such as recent work learning atomic relations and ontologies (Banko et al., 2007b). I believe my algorithm for template-based IE complements this work by focusing on template structure.

Finally, my approach to the MUC-4 task is a pipelined solution. Starting from the raw text, I have split the algorithm into two separate stages: template induction and entity extraction. Further, since the MUC-4 corpus is so small, I have a third Information Retrieval stage that is embedded within the template structure induction stage. The benefits of this pipeline include flexibility in development and more transparency in identifying errors. However, there are also several parameters in each stage that need to be set and tuned to the task. I believe the IR parameters are quite robust to change, and did not intently focus on improving this stage. However, the initial clustering of event words and the clustering of slots during template induction involves a couple clustering steps with parameters that control stopping conditions and word filtering. I optimized some of these, as described above in this chapter, and chose others by hand through visual inspection of example training output. All learning algorithms require parameters, including generative models like LDA, but future work will need to focus on removing as many of these as possible to allow algorithmic transfer to new domains and genres.

The templates learned in this chapter are equivalent to narrative event schemas. The algorithm was adapted to fit an IE domain, but the knowledge structures are the same. Whereas Chapters 2 and 3 presented intrinsic evaluations of narrative schemas, this chapter’s results present an evaluation of the usefulness of learning narrative schemas in a different NLP application. I assumed less knowledge at the
start than previous work; I allowed narrative schemas to fill in the knowledge gap; and completed the extraction task with comparable performance to previous work.
Chapter 7

Conclusions

This dissertation explored a new knowledge representation for characterizing situations in the world: the Narrative Event Schema. I presented approaches to learning this knowledge from unlabeled text, classifiers with new temporal reasoning components, and several evaluations, including its integration into an end-user information extraction application. I believe this work is an important contribution that connects the richness of historical NLP models with the robustness and flexibility of today’s statistical learning models.

7.1 Contributions

The central contribution of this dissertation is the Narrative Schema representation. Chapters 2 and 3 described how it uniquely captures events, entities, and the syntactic constraints of the roles the entities fill in one single representation. I described the core representation as a narrative chain: a set of events that are semantically connected by a single actor, called the protagonist. I showed how a protagonist is defined by its syntactic roles, and how chains can be extended into the full narrative schema representation by characterizing all actors in the narrative. Finally, I showed how the statistical relationships between the events and entities fulfill the modern interpretation of Schankian scripts, and enable a unique approach to learning this knowledge from open-domain text.
Chapter 4 then described my contributions to the fields of event ordering and temporal reasoning. I described the first supervised approach to event-event ordering that built its features from raw text, and the results are still state-of-the-art today. I also described the first model of temporal event order that enforces global transitivity constraints on the local event-event decisions, maintaining consistency across the entire document. I investigated reasons such a global model can be difficult, and presented results improving over a local classifier. I concluded the temporal reasoning part of this dissertation in Chapter 5 and showed how to impose event orderings over learned schemas.

Other contributions of this dissertation include several new evaluations for structured event representations, and the results from these evaluations using my database of learned narrative schemas. I created a new evaluation called the narrative cloze by adapting the Cloze Test from linguistics to event relationships. It evaluated the predictive power of the learned events. I also evaluated both precision and recall of my learned schemas by comparing against the FrameNet database, and a coverage experiment over newspaper articles. Finally, I evaluated the temporal order through a consistency evaluation that permutes the true order of a document and relies on the model to predict the original order. Through these four intrinsic evaluations, I showed the high level of precision of my schema database, and its appropriate level of recall against real-world data.

Finally, I concluded this dissertation in Chapter 6 with an extrinsic, application-specific evaluation that applied my schema learning process to a completely different field in NLP: information extraction. I mapped my narrative schema representation to the templates in a common template-based information extraction task, and integrated my learning process into that framework. All previous work on this task assumed knowledge of hand-coded templates and focused solely on extracting fillers for the templates. This dissertation is first to approach this task without templates, and instead learn them in the form of narrative schemas. Once learned, I extracted their fillers and ultimately achieved extraction results that approached the performance of supervised learners. My narrative schema framework thus offers a new direction for unsupervised extraction.
7.2 Future Work

The space of learning how events and entities interact is very much unexplored. This dissertation is one of the first explorations to learn a representation that includes both events and entities in one structure. As such, several open questions remain. I highlight a few here.

Can we build a hierarchy of schemas to jointly inform schema structure? I have shown how to learn a flat hierarchy of schemas. Each schema is a self-contained set of events, but there are no semantic connections across schemas except for perhaps word overlap. Many schemas, however, include entities that fill the same semantic role. Using the IE domain from Chapter 6 as an example, the four target situations all include a person who is a perpetrator. These four situations should be under a single generalized schema for crime or terrorist event. They then inherit the role from the parent, and can jointly inform each other during learning and extraction. Learning hierarchies is an open problem in machine learning, and hierarchical schemas will require new models of selectional preferences that can detect role-bearing words (e.g., suspect) when they appear in text.

Can we encode schemas recursively? One of the results of this dissertation is that schemas vary in the granularity of events that they describe. The granularity is currently controlled by the corpus, however, and the algorithm learns the granularity contained in the text itself. For instance, I learned a “life schema” from obituaries that includes broad life events like born, graduate, retire, die. I also learned schemas that represent more specific events that characterize situations like a graduation. Future work that can detect this granularity and embed schemas within schemas holds promise for advanced reasoning systems and better document understanding.

How do we separate synonymy from narrative relations? The majority of unsupervised learning in NLP focuses on clustering words and patterns that are synonymous. By using a protagonist, among other things, I have shown how to learn all events in a situation, not just synonymous relations. Many synonyms are implicitly captured by the protagonist as well, so the resulting schemas contain a mix of both synonyms (e.g., plant, place) and broader narrative relations (e.g., plant, explode). Making
this distinction within a schema may lead to more precise learning algorithms that can first cluster synonyms, and then discover narrative relations. I believe this will require a mixture of approaches that utilize distributional approaches to synonymy, and protagonist-based work.

Can we build a unified induction and extraction framework for template-based information extraction? My approach to information extraction is a pipelined approach that first optimizes template structure induction, and then optimizes the entity extraction stage. There is some feedback in adjusting induction parameters based on extraction performance, but the interaction between stages is not significant. The most obvious area for improvement is reasoning over induced slots and their per-document extraction. For instance, induced slots should extract entities in most of the template’s document matches. If a slot rarely extracts instances, this could be used as negative reinforcement in the induction process. In contrast, if a template’s matched documents contain repeated mentions of words that are not being extracted, these words can be used as indicators of a new slot. Generative models offer tools to build such a joint model, but techniques to make inference tractable over the topics, their events, syntactic slots, and dependencies like the protagonist, are challenges to overcome. Other models not yet explored that can feedback extraction results to induction may perform more efficiently.

Finally, how does narrative knowledge assist other NLP applications? I explored one particular application, information extraction, and showed how schema learning can transform the task from supervised to largely unsupervised. Many of today’s NLP tasks focus on sentence-level applications, such as parsing, named entity recognition, machine translation, etc. As the field moves to full document understanding and incorporates more of the context inherent in a document, I believe models like narrative schemas will offer a deeper representation of semantics to better model the context. I look forward to learning and utilizing rich models of event structure across a host of applications.
Bibliography


He, Shan, and Daniel Gildea. 2006b. Self-training and co-training for semantic role labeling: Primary report. Technical Report 891, University of Rochester.


Paul, Michael, Roxana Girju, and Chen Li. 2009. Mining the web for reciprocal relationships. In *Conference on Computational Natural Language Learning*.


