Extraction of Event Structures from Text

Jun Araki

CMU-LTI-18-007

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA
www.lti.cs.cmu.edu

Thesis Committee:
Teruko Mitamura (Chair), Carnegie Mellon University
Eduard Hovy, Carnegie Mellon University
Graham Neubig, Carnegie Mellon University
Luke Zettlemoyer, The University of Washington

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

Keywords: event structures, event detection, event coreference resolution, subevent detection, bidirectional long short-term memory, distant supervision, multi-task learning, computer-assist language learning

Abstract

Events are a key semantic component integral to information extraction and natural language understanding, which can potentially enhance many downstream applications. Despite their importance, they have received less attention in research on natural language processing. Salient properties of events are that they are a ubiquitous linguistic phenomenon appearing in various domains and that they compose rich discourse structures via event coreferences, forming a coherent story over multiple sentences.

The central goal of this thesis is to devise a computational method that models the structural property of events in a principled framework to enable more sophisticated event detection and event coreference resolution. To achieve this goal, we address five important problems in these areas: (1) restricted domains in event detection, (2) data sparsity in event detection, (3) lack of subevent detection, (4) error propagation in pipeline models, and (5) limited applications of events. For the first two problems, we introduce a new paradigm of open-domain event detection and show that it is feasible for a distant supervision method to build models detecting events robustly in various domains while obviating the need for human annotation of events. For the third and fourth problems, we show how structured learning models are capable of capturing event interdependencies and making more informed decisions on event coreference resolution and subevent detection. Lastly, we present a novel application of event structures for question generation, illustrating usefulness of event structures as inference steps in reading comprehension by humans.

For my family.

Acknowledgments

I have been greatly fortunate to have interacted with excellent individuals who have had a profound impact on my Ph.D. training. In this note of acknowledgment, I would like to express my heartfelt gratitude to them and describe how supportive the interactions have been for the challenging process of my pursuing a Ph.D.

First and foremost, I would like to express my sincere appreciation to my advisor, Teruko Mitamura. This thesis would not have been possible without her continuous guidance and support. She was always approachable whenever I needed guidance. Her enlightning guidance and keen insight into language have helped me make changes in the right direction when I drifted off course.

I would like to give my deep gratitude to Eduard Hovy. Through his valuable comments and discussions in our group and project meetings, I learned from him scientific mind in research on natural language and the importance of being a thoughtful researcher.

I would also like to express my sincere thanks to my other thesis committee members, Graham Neubig and Luke Zettlemoyer. I sincerely thank Graham Neubig for his valuable comments on this dissertation work. Discussions with him helped me a lot in clarifying methodological novelties in the thesis. I am greatly grateful to Luke Zettlemoyer for his insightful advice. I like his way of putting research problems in a broad context and situating them via connections to others.

I am grateful to faculty members and staff members at Carnegie Mellon University. Kemal Oflazer gave an opportunity to expand my research expertise to computer-assisted language learning. I am equally grateful to my colleagues for helpful discussions and encouragement: Hideki Shima, Zhengzhong Liu, William Yang Wang, Pradeep Dasigi, Zi Yang, Di Wang, Leonid Boytsov, Sreecharan Sankaranarayanan, Dheeraj Rajagopal, Danish Pruthi, and Evangelia Spiliopoulou. I thank Yukari Yamakawa and Susan Holm for performing annotation used in this thesis. I am also thankful to my mentors at IBM Research, Rick Lawrence and Abhishek Kumar, for their mentorship during my internship with IBM Research.

I gratefully acknowledge the funding sources that made my Ph.D. study possible. This work is supported in part by a Funai Overseas Scholarship from the Funai Foundation for Information Technology, IBM Ph.D. Fellowships, DARPA grant FA8750-12-2-0342 funded under the Deep Exploration and Filtering of Text (DEFT) program, U.S. Army Research Office (ARO) grant W911NF-14-1-0436 under the Reading, Extraction, and Assembly of Pathways for Evidentiary Reading (REAPER) program, and grant NPRP-08-1337-1-243 from the Qatar National Research Fund (a member of the Qatar Foundation).

Lastly, I would like to thank my family and friends for their unconditional love and continuous support.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Definition of Events . . . 2
  - 1.2.1 Linguistic Perspectives . . . 2
  - 1.2.2 Computational Perspectives . . . 4
  - 1.2.3 Our Definition of Events . . . 4
  - 1.2.4 Other Topics Related to Events . . . 8
- 1.3 Definition of Event Coreference . . . 9
  - 1.3.1 Full Identity of Events . . . 10
  - 1.3.2 Partial Identity of Events . . . 11
  - 1.3.3 Other Topics Related to Event Coreference . . . 12
- 1.4 Problem Statements . . . 13
  - 1.4.1 Restricted Domains in Event Detection . . . 13
  - 1.4.2 Data Sparsity in Event Detection . . . 13
  - 1.4.3 Lack of Subevent Detection . . . 15
  - 1.4.4 Error Propagation in Pipeline Models . . . 16
  - 1.4.5 Limited Applications of Events . . . 16
  - 1.4.6 Summary . . . 17
- 1.5 Goal and Contributions . . . 18
- 1.6 Thesis Outline . . . 19

2 Datasets and Evaluation **21**
- 2.1 Datasets . . . 21
  - 2.1.1 ACE 2005 . . . 21
  - 2.1.2 ERE and TAC KBP . . . 23
  - 2.1.3 Intelligence Community Corpus . . . 25
  - 2.1.4 ProcessBank . . . 26
  - 2.1.5 Simple Wikipedia Corpus (SW100) . . . 26
- 2.2 Evaluation . . . 28
  - 2.2.1 Event Detection . . . 28
  - 2.2.2 Full Event Coreference Resolution . . . 30
  - 2.2.3 Proposed Evaluation for Partial Event Coreference Resolution . . . 30
- 2.3 Related Work . . . 39
  - 2.3.1 Human Annotation of Event Datasets . . . 40

2.3.2 Tree Similarity . . . 41
2.4 Summary . . . 42

3 Event Detection **43**
3.1 Closed Domain Event Detection . . . 43
3.2 Open Domain Event Detection . . . 44
3.3 Event Argument Detection with Semantic Parsing . . . 44
3.4 Supervised Closed Domain Event Detection . . . 45
3.4.1 Conditional Random Fields . . . 45
3.4.2 Bidirectional Long Short-Term Memory . . . 46
3.4.3 Realis Classification . . . 50
3.4.4 Experiments and Discussions . . . 51
3.5 Distantly-supervised Open Domain Event Detection . . . 55
3.5.1 Training Data Generation . . . 56
3.5.2 Enhancements with Wikipedia . . . 57
3.5.3 Learning for Event Detection . . . 60
3.5.4 Experiments and Discussions . . . 60
3.6 Related Work . . . 64
3.6.1 Event Detection . . . 64
3.6.2 Event Argument Detection . . . 67
3.6.3 Semi-supervised and Distantly-supervised Learning in NLP . . . 68
3.7 Summary . . . 69

4 Event Coreference Resolution **71**
4.1 Full Event Coreference Resolution . . . 71
4.1.1 Heuristic Approaches . . . 71
4.1.2 Latent Antecedent Tree Model . . . 72
4.1.3 Neural Event Coreference Model . . . 73
4.1.4 Experiments and Discussions . . . 75
4.2 Detecting Subevent Structures . . . 76
4.2.1 Subevent Structures . . . 77
4.2.2 Event Relation Learning . . . 78
4.2.3 Subevent Detection . . . 78
4.2.4 Experiments and Discussions . . . 81
4.3 Related Work . . . 83
4.3.1 Full Event Coreference Resolution . . . 83
4.3.2 Subevent Detection . . . 84
4.4 Summary . . . 85

5 Joint Modeling for Event Detection and Event Coreference Resolution **87**
5.1 Event Interdependencies via Event Coreference . . . 87
5.2 Joint Modeling with Feature-based Models . . . 89
5.2.1 Event Graph Learning . . . 90
5.2.2 Joint Decoding . . . 91

5.2.3 Experiments and Discussions . . . 93
5.3 Joint Modeling with Neural Models . . . 95
5.3.1 Joint Decoding . . . 95
5.3.2 Joint Training . . . 95
5.3.3 Experiments and Discussions . . . 97
5.4 Related Work . . . 99
5.4.1 Joint Learning of Feature-based Models in NLP . . . 99
5.4.2 Joint Learning of Neural Network Models in NLP . . . 99
5.5 Summary . . . 99

6 Applications of Events **101**
6.1 Question Generation . . . 101
6.1.1 Generating Questions using Coreferences and Paraphrases . . . 102
6.1.2 Evaluation of Generated Questions . . . 105
6.1.3 Experiments and Discussions . . . 106
6.2 Related Work . . . 107
6.3 Summary . . . 109

7 Conclusion **111**
7.1 Future Work . . . 112

A Annotation Guidelines for Open-Domain Event Nuggets **115**
A.1 Introduction . . . 115
A.2 Principles of Event Annotation . . . 115
A.2.1 Semantic Perspective: Eventualities . . . 115
A.2.2 Syntactic Perspective: Event Nuggets . . . 116
A.3 General Rules . . . 116
A.4 Annotating Eventive Verbs . . . 117
A.5 Annotating Eventive Nouns . . . 121
A.6 Annotating Eventive Adjectives . . . 123
A.7 Annotating Eventive Adverbs . . . 125

Bibliography **127**

List of Figures

1.1 Classification of eventualities by Bach (1986). 3

2.1 An example of expert annotation on a paragraph in ProcessBank, visualized with a modified version of BRAT (Stenetorp et al., 2012). In this example, a purple arrow with 'Coref' represents an entity coreference, an orange one with 'Same' a full (event) coreference, and a red one with 'Super' an opposite of a subevent relation. A green arrow with 'Agent' and a blue one with 'Theme' represent an agent argument and and a theme argument of an event mention, respectively. . . . 27
2.2 Examples of subevent and membership relations. An arrow and a dashed arrow represent a subevent and a membership relation with the direction from a parent to its subevent and member, respectively. For example, E55 is a subevent of E57. Additionally, a straight line represents full coreference. 31
2.3 A conceptual subevent tree constructed from the full coreference and subevent relations in Figure 2.2. 33
2.4 Conversion from a forest to a single tree with an additional dummy root. 36
2.5 Score comparison among MUC_p, $BLANC_p$, and $NSTM_p$. The number of correct links increases from singletons to the perfect output (the gold standard) one by one. 38
2.6 Score comparison among MUC_p, $BLANC_p$, and $NSTM_p$. The number of incorrect links increases from the perfect output to a single tree merging all trees one by one. 38

3.1 A character-level convolutional neural network (CharCNN). 49
3.2 A high-level architecture of our realis classification model. 50
3.3 An overview of our distantly-supervised open-domain event detection. 55
3.4 The bidirectional LSTM model with a self-attention mechanism. 59
3.5 Performance of the event detection model on SW100 with respect to the number of training examples generated from SemCor. 62

4.1 A high-level architecture of the first subnetwork to construct event representation. 74
4.2 A high-level architecture of the second subnetwork to compute antecedent scores. 74
4.3 An example of subevent structure. An arrow represents a subevent parent-child relation with the direction from a parent to its subevent. A line represents a subevent sister relation between subevents under the same parent. 77

4.4 Excerpts from narrative schemas relevant to events in the IC domain. In each schema, the first line shows the overall score for that schema, and the third shows the individual verb scores, aligned with verbs in the second. 80
4.5 Excerpts from the subevent ontology tree constructed from the training data set. The numbers in each node show a frequency of the headword of an event mention and its ratio (percentage) to the total number of occurrences of event mentions, which is 350. The tree shows subevent parents in the first level and subevent sisters in the second level. 80
4.6 An example subevent structure detected in each stage. E82 is a missing event that the system fails to detect. 82
4.7 Parent selection from subevent sisters. 83

5.1 An event structure for Example (75) in the newswire domain. Dashed straight lines and arrows represent event types and event arguments, respectively. Solid straight lines and arrows represent event coreferences and subevents, respectively. 89
5.2 A neural architecture for joint training of event detection and event coreference resolution. 96

6.1 A paragraph with annotation of events, entities and their relations in ProcessBank. A 'Same' link means event coreference, whereas a 'Coref' link means entity coreference. 104
6.2 An example text to generate a question using an entity coreference. 104
6.3 An example text to generate a question using a paraphrase. 105

List of Tables

1.1 Definition of terminology regarding events. 7

1.2 A slot-filling table representation of the buying event in Example (13). 8

1.3 Definition of terminology regarding event coreference. 12

1.4 Comparison between reported performances of state-of-the-art systems for event trigger detection on ACE 2005. 'P' and 'R' stand for precision and recall, respectively. 14

1.5 Comparison between reported performances of state-of-the-art systems for event nugget detection on TAC KBP 2015. The first five systems are the top five official submissions to the TAC KBP 2015 Event Nugget track. 14

1.6 A structured overview of this thesis with respect to the problems stated in Section 1.4. 20

2.1 Datasets and associated tasks in which they are used. 21

2.2 8 event types and 33 event subtypes defined in the ACE 2005 corpus. 22

2.3 Statistics of the ACE 2005 corpus. In (a), triggers, arguments, and clusters denote event triggers, event arguments, and event (coreference) clusters, respectively. In (b), Prn, Adj, and Adv denote pronoun, adjective, and adverb, respectively. Parentheses show ratios with respect to the percentage. 22

2.4 7 event types and 18 event subtypes defined by the TAC KBP Event track in 2016 and 2017. 23

2.5 Statistics of event nugget datasets in LightERE, Rich ERE and TAC KBP. In the text type field, NW and DF refer to newswire and discussion forums, respectively. Parentheses show the numbers when event nuggets are reduced to the 18 event types shown in Table 2.4. Event clusters include both clusters with singletons and ones with multiple coreferential event nuggets, as defined in Section 1.3.1. The Rich ERE 2016 dataset contains discussion forum data only, and each thread is splitted into one or more small units, called CMP; we regard one CMP as one document in the dataset because event coreference is annotated within a CMP unit. 24

2.6 Statistics of the TAC KBP corpus. Parentheses show ratios with respect to the percentage. In (a), NW and DF refer to newswire and discussion forums, respectively. The row of "# multi-tagged spans" shows the number of spans annotated with two or more event types (double tagging). In (b), Prn, Adj, and Adv denote pronoun, adjective, and adverb, respectively. In (d), we found that 11 event nuggets from the TAC KBP 2014 dataset were not annotated with realis (missing gold standard realis values). . . . 25
2.7 Statistics of the Intelligence Community (IC) corpus. . . . 26
2.8 Statistics of the ProcessBank corpus. . . . 26
2.9 Corpus statistics of SW100. Percentages (%) are shown in parentheses. . . . 28
2.10 Examples of a system response against a gold standard partial coreference. Each event tree is shown in the bold font and in the Newick standard format with parentheses. . . . 39

3.1 Features of CRF models for event detection. . . . 46
3.2 Statistics of our datasets. . . . 51
3.3 The performance of the LSTM model on the test dataset, with respect to different settings of the dimension d_h of the hidden state. . . . 52
3.4 Results of event trigger detection on our test data of the ACE 2005 corpus. . . . 52
3.5 Performance of event detection with respect to spans. . . . 53
3.6 Performance of event detection with respect to types (span+type). . . . 53
3.7 Performance of our realis classifiers with respect to the F1 score for each realis value. The F1 score for 'Overall' is computed with micro F1. . . . 54
3.8 The confusion matrix of realis classification by the BLSTM+CharCNN model. 'A', 'G' and 'O' stand for ACTUAL, GENERIC and OTHER, respectively. . . . 54
3.9 Performance of event detection with respect to realis (span+realis). . . . 55
3.10 Overall performance of event detection (span+type+realis). . . . 55
3.11 Examples of WordNet glosses in D_+ and D_-. . . . 59
3.12 Accuracy of gloss classifiers on the datasets from WordNet and Wikipedia. The stars indicate statistical significance compared to the GC-BLSTM model (*: $p < 0.05$; **: $p < 0.005$) based on McNemar's test. . . . 61
3.13 Performance of the rule-based event detectors on SW100. . . . 61
3.14 Results of event detection. . . . 62
3.15 Detailed performance of DS-BLSTM. . . . 63
3.16 Noticeable errors of our training data generation. . . . 64
3.17 Comparison between reported performances of event trigger detection on the same ACE 2005 test set used in (Ji and Grishman, 2008). 'P' and 'R' stand for precision and recall, respectively. . . . 65
3.18 Comparison between reported performances of state-of-the-art systems for event nugget detection on TAC KBP 2015. The first five systems are the top five official submissions to the TAC KBP 2015 Event Nugget track. . . . 67

4.1 Distributions of event types and realis values over event coreference clusters in the TAC KBP corpus. 'A', 'G' and 'O' stand for ACTUAL, GENERIC, and OTHER, respectively. . . . 72
4.2 The official results (F1 scores) of our system on the event hopper coreference task. 76
4.3 Performance (F1 scores) of event coreference resolution in the TAC KBP 2017 dataset. 'Top N' represents the Nth-ranked system reported in the official results. 76
4.4 A list of the features for our event relation learning. A number within parentheses in each feature group shows how many features belong to that group. . . . 79
4.5 BLANC scores gained in the first stage. . . . 82
4.6 BLANC scores gained in the second stage. . . . 82

5.1 A list of features for event trigger identification. . . . 90
5.2 A list of features for event coreference resolution. . . . 91
5.3 Results (F1 scores) of event coreference resolution. . . . 94
5.4 Results of event trigger identification. 'Baseline' refers to the first stage of our baseline. . . . 94
5.5 Performance of event detection with respect to types (span+type). The star (*) indicate statistical significance compared to the BLSTM-MLC model at $p < 0.05$, based on a two-tailed paired t-test. . . . 97
5.6 Overall performance of event detection (span+type+realis). The star (*) indicate statistical significance compared to the BLSTM-MLC model at $p < 0.05$, based on a two-tailed paired t-test. . . . 97
5.7 Performance (F1 scores) of event coreference resolution in the TAC KBP 2017 dataset. 'Top N' represents the Nth-ranked system reported in the official results. The stars (*) indicate statistical significance compared to the NEC model at $p < 0.05$, based on a two-tailed paired t-test. . . . 98

6.1 Question patterns and templates using event coreference, entity coreference, and paraphrases. In question patterns, En denotes an event trigger, and Enn an entity mention. A straight line denotes a coreference link, a dashed arrow an 'Agent' relation, and a straight arrow a relation which is 'Cause', 'Enable' or 'Result'. An event clause in question templates is defined as a text span including an event trigger and its arguments. . . . 103
6.2 The performance comparison in question generation. Numbers in grammatical correctness and answer existence are average ratings, and lower is better. Numbers in inference steps are average inference steps, and higher is better. . . . 107
6.3 Average ratings of answer correctness in 200 questions. Lower numbers are better. Scores range from 1-3, with 1 a correct answer. . . . 107

List of Algorithms

1 Extended simple tree matching for unordered trees. 36
2 Structured perceptron. 73
3 Joint decoding for event triggers and coreference with beam search. 92

Chapter 1

Introduction

We give an introduction of this thesis by starting with our motivation to study events in Section 1.1. We discuss the definition of events and their coreferences in Section 1.2 and Section 1.3, respectively. We then state five important problems in state-of-the-art work on events and their coreferences in Section 1.4. We describe the goal and contributions of this thesis in Section 1.5. Section 1.6 shows the outline of this thesis.

1.1 Motivation

An overwhelming amount of unstructured text, such as newspaper articles and biomedical research papers, is accessible in the world today, and the amount is continuing to increase at an unprecedented speed. Consequently, any single person cannot consume all the information related to a topic or even a single entity in order to obtain a complete picture of it. Information extraction (IE) technologies have been studied to alleviate the difficulty. The key information that IE is aimed to capture is semantic information that **events** convey: who did what to whom where and when. For instance, if a large earthquake occurs in a certain country, its government will probably need to analyze a potentially large amount of text, such as newspaper articles, to collect important factual information about the earthquake as accurately and quickly as possible, including where and when it happened, and how many people are affected. This analysis is integral to the government's next action plans for the earthquake. Looking over the world, we see countless events of different granularities happening in our daily lives: terrorist attacks in some cities, presidential elections in some countries, releases of new high-tech products, outbreaks of a virus in some rural areas, discoveries of novel protein-protein interactions relevant to a certain type of cancer, and so forth. The increase of textual information at an unprecedented speed accelerates our need for event-level analysis of text to meet the goal of IE.

However, state-of-the-art event processing technologies is still far from the level necessary to meet the goal. More effort on detecting events and resolving relations between events accurately is crucial to provide underlying essential representations of text in meaningful ways, thereby enabling us to absorb and disseminate our knowledge more efficiently and effectively in different domains. This gap between the current limitation of the state-of-the-art event processing technologies and the need for the semantic analysis of text gives us a fundamental motivation to

study events. An event is often defined as something that happens[1]. Their salient property among other linguistic phenomena is that they compose rich semantic argument structures and discourse structures. On the one hand, they involve various participants and attributes locally, often within a sentence, and form a semantic (argument) structure: who did what to whom where and when. On the other hand, they relate to each other in different ways globally, often across sentences, and forms a discourse structure to tell us a coherent story about a certain subject. We believe that both views are indispensable to address the limitation of the state-of-the-art IE technologies and eventually advance the understanding of natural language.

In light of the rich semantic argument and discourse structures of events, it is not surprising that events can be utilized in a large variety of natural language processing (NLP) applications. In fact, the semantic and discourse structures of events have been already utilized by a host of NLP applications, such as automated population of knowledge bases (Ji and Grishman, 2011), information extraction (Humphreys et al., 1997), topic detection and tracking (Allan, 2002), question answering (Bikel and Castelli, 2008), text summarization (Li et al., 2006), textual entailment (Haghighi et al., 2005), contradiction detection (de Marneffe et al., 2008), and stock market prediction (Ding et al., 2014). This wide range of NLP applications of events also illustrate the fact that events are a core component for text analysis. We explicate those existing applications of events in Chapter 6.

1.2 Definition of Events

It is difficult to provide a clear-cut definition of events because they are a conceptually ambiguous notion. Hence, various definitions of events could be possible, depending on a particular purpose or standpoint. In fact, different researchers have defined events from different perspectives. In this section, we provide a discussion on the definition of events from two perspectives: linguistics (Section 1.2.1) and computation (Section 1.2.2). We then describe our definition of events using a combination of the notions of eventualities (Bach, 1986) and event nuggets (Mitamura et al., 2015b).

1.2.1 Linguistic Perspectives

The definition of events has a deep connection with linguistic studies on verbs. In theoretical linguistics, one can go back to the work of Vendler (1957) on aspectual classification of verbs. Vendler (1957) classifies verbs into four categories: states, activities, accomplishments, and achievements. Recent work extends Vendler's classification by not restricting it to verbs, and introduces the tripartite distinction into states, processes and events (Mourelatos, 1978; Carlson, 1981; Bach, 1986). In particular, Bach (1986) introduces the notion of **eventualities**, which are a broader notion of events and define the three components on the basis of durativity and telicity (Moens and Steedman, 1988; Pulman, 1997):

- **states**: a class of notions which are durative and changeless, e.g, want, own, love, resemble
- **processes**: a class of notions which are durative and atelic, e.g., walking, sleeping, raining

[1] We give our formal definition of events in Section 1.2.3.

- **actions**[2]: a class of notions which are telic or momentaneous happenings, e.g., build, walk to Boston, recognize, win, arrive, clap

Durativity concerns whether a notion has a duration, and telicity means that a notion has an explicit condition of termination. Eventualities first distinguish states from non-states, recognizing that states do not involve any changes. Non-states are further divided into processes and events, based on the notion of telicity. Figure 1.1 shows the hierarchical classification of eventualities.

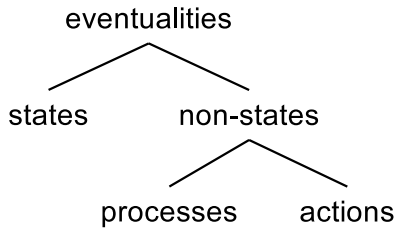


Figure 1.1: Classification of eventualities by Bach (1986).

On the other hand, the construction of a lexical resource related to verbs often involves the definition of events in the process of taxonomizing words or concepts. Every verb taxonomy provided by a particular lexical resource or extracted from it in some manner can be viewed as an extensional definition of events. WordNet (Miller et al., 1990) is a lexical database that organizes over 150,000 English words (over 11,000 verbs) by grouping them into sets of synonyms, and connect them via a number of relations. It defines events as "something that happens at a given place and time."[3] FrameNet (Baker et al., 1998) is a lexical database based on a theory of meaning called frame semantics (Fillmore, 1976) and comprises around 1,200 semantic frames, each of which denotes a coherent structure of related concepts. It provides the definition of frame 'Event' as "An event takes place at a place and time." Levin (1993) classifies 3,100 verbs into a hierarchy of 47 top-level classes and 193 second- and third-level classes, according to syntactic signatures based on alternations. VerbNet (Kipper et al., 2008) organizes verb classes extending Levin's through refinement and addition of subclasses, comprising 274 top-level classes and 3,769 lemmas. For each verb class, VerbNet provides a syntactic description and semantic predicates with a temporal function in a way akin to the event nucleus described above. PropBank (Palmer et al., 2005) is a corpus of sentences which annotate verbal propositions and their arguments, being aimed at providing a broad-coverage (3,257 verbs) of the alternation in the syntactic realization of semantic arguments. In a similar vein, NomBank (Meyers et al., 2004) provides a set of human-annotated argument structures evoked from 4,704 nouns in the PropBank. OntoNotes (Pradhan et al., 2007a) is a corpus of integrated annotation of multiple levels of the shallow semantic structure in text. It provides the proposition structure of both verbs and nouns in PropBank while clarifying the difference between nominalization senses and eventive senses of nouns.

Other linguistic studies analyze the underlying semantic structure of events, paying attention to semantic constraints in sentences to distinguish between events, extended events, and states (Chung and Timberlake, 1985; Pustejovsky, 2000). For instance, Chung and Timberlake

[2] Bach (1986) uses the term 'events' to refer to this class. In this work, we use 'actions' instead for clarification.

[3] http://wordnetweb.princeton.edu/perl/webwn?s=event

(1985) state "an event can be defined in terms of three components: a predicate; an interval of time on which the predicate occurs, ... and a situation or set of conditions under which the predicate occurs."

1.2.2 Computational Perspectives

Researchers have also given different definitions of events from different computational perspectives. In information retrieval (IR), prior works on topic detection and tracking treat an event as a narrowly defined topic which can be categorized as a set of related documents (Allan et al., 1998). This document-level definition specifies broader events that may include happenings and their causes, consequences, or even more extended effects.

In contrast, previous works on information extraction (IE) define events with finer granularity. In the seventh Message Understanding Conference (MUC-7), an event represents a structured template that relates an action to its participants, times, places, and other entities involved (Marsh and Perzanowski, 1998). This sentence-level definition conforms with what Filatova and Hatzivassiloglou (2003) define as an atomic event. TimeML (Pustejovsky et al., 2003) is a rich specification language for event and temporal expressions in natural language text. TimeML defines an event as "a cover term for situations that happen or occur" and a predicate "describing states or circumstances in which something obtains or holds true". The annotation guidelines of the Automatic Content Extraction (ACE) program (LDC, 2005) defines an event as a specific occurrence of something that happens involving participants, often described as a change of state. The ECB+ corpus defines an event as a combination of four components: an action, a time, a location, and a participant (Cybulska and Vossen, 2014). From a relation perspective, an event can be seen as an n-ary relation with an event expression its core, being linked to its arguments. (McDonald et al., 2005) leverages this perspective to formalize biomedical event extraction as a problem of relation extraction.

1.2.3 Our Definition of Events

In this thesis, we consider both the linguistic and computational perspectives, and follow the definition of eventualities by Bach (1986) and the definition of events used in the IE community. Using the notion classes introduced in Section 1.2.1, we define an **event** to be *an eventuality which involves zero or more of participants, attributes, or both*. The first thing to note is that this definition formalizes an event not as a concrete mention in text but as an abstract notion independent of any particular text. To illustrate this point, let us consider the following example sentence:

(1) John bought a novel at a bookstore yesterday.

In this case, the event that we define is not the sentence of Example (1) but an abstract notion of the buying event which the sentence refers to. In other words, the sentence is a textual mention that refers to the buying event. We call such mention an **event mention**, which we will define formally below.

Following the definition of events, we define several event components in text annotation. The most representative word for the event in Example (1) is 'bought' because it expresses the

event most clearly. We call such main word an **event trigger**, following the definition by the ACE program. An event trigger is a single word or a multi-word phrase, but in reality the vast majority (approximately 95%) of ACE event triggers consist of a single token. This dominance of single-token ACE triggers seems rather restrictive presumably since in general there are a substantial number of multi-token event expressions, such as verb phrases (e.g., 'look into' and 'take a walk') and compound proper nouns (e.g., 'the 2004 Indian Ocean earthquake' and 'Rio Olympics'). In addition, some single-token event triggers can express only partial semantics of an event. We give two examples to illustrate this point:

(2) The university will send emails to all students tomorrow.

(3) The hurricane left 20 people dead.

The event triggers in Example (2) and Example (3) are 'emails' and 'dead', respectively, because they are a main word that expresses the corresponding event most clearly. However, 'emails' and 'dead' seem inadequate as a trigger, not just due to their syntactic form but also because their lexical semantics is not expressive enough to semantically cover the events that they are trying to tell us. To address these issues, Mitamura et al. (2015b) propose a notion of **event nugget** as a semantically meaningful unit that expresses an event. An event nugget is a single word or a multi-word phrase, but unlike an event trigger, it can be discontinuous words. For instance, the event nugget in Example (2) is "sent emails", and the one in Example (3) is "left ... dead". As seen in these examples, an event nugget is the largest extent of text that expresses an event, whereas an event trigger is the smallest extent for the same purpose.

In Example (1), the buying event involves two participants (i.e., John and a novel) and two attributes (i.e., a bookstore and yesterday). We define an **event argument** as a participant or an attribute involved in an event, following the definition by the ACE program. As seen in the example, event arguments have several roles for an event trigger. In this thesis, we model an event by using the following four event arguments:

- **Agent**: a participant that causes or initiates an event.
- **Patient**: a participant upon whom an event is carried out.
- **Time**: an attribute that specifies when an event happens.
- **Location**: an attribute that specifies where an event happens.

Note that this definition is similar to the semantic role definition given by PropBank. In Example (1), John is an agent, a novel is a patient, yesterday is a time, and a bookstore is a location of the event. Note that participants collectively refer to agents and patients, and attributes collectively refer to times and locations in our definition. Some events do not have some of the participants and attributes defined above. For instance, Example (3) has an agent (i.e., the hurricane) and a patient (e.g., 20 people), but does not have a patient and a time. We now define an **event mention** as a mention in text that describes an event, and includes an event trigger (or event nugget) and event arguments. An event mention is typically a clause as seen in Example (1), Example (2) and Example (3).

To refer to a particular event expression that we have introduced before, let E$\boldsymbol{i}$ denote a unique identifier for the event expression where i is a positive integer. Note that this identifier simply enumerates all event instances, and does not denote whether a referred event expression

is identical to previous ones. We insert it to text with parentheses right after its corresponding event trigger or nugget in bold face:

(4) John **bought**(E1) a novel at a bookstore yesterday.

(5) The university will **send emails**(E2) to all students tomorrow.

(6) The hurricane **left** 20 people **dead**(E3).

E1 is an event trigger in Example (4), whereas E2 and E3 are an event nugget in Example (5) and Example (6), respectively. If we want to refer to a particular event mention, we say "the event mention based on E1" or "the event mention of E1". If we want to specify event arguments in addition to the event expression identifier, we use underlines as follows:

(7) John **bought**(E4) a novel at a bookstore yesterday.

(8) The university will **send emails**(E5) to all students tomorrow.

(9) The hurricane **left** 20 people **dead**(E6).

This notation only specifies event argument spans without any argument roles. To specify argument roles additionally, let Ei-r denote an event argument of Ei with role r. We insert the notation underneath the underlined event argument as follows:

(10) John **bought**(E7) a novel at a bookstore yesterday.
E7-A0 E7-A1 E7-LOC E7-TMP

(11) The university will **send emails**(E8) to all students tomorrow.
E8-A0 E8-A1 E8-TMP

(12) The hurricane **left** 20 people **dead**(E9).
E9-A0 E9-A1

For argument role r, we use a notation similar to the semantic role notation by PropBank: A0 for agent, A1 for patient, TMP for time, and LOC for location. Note that the above argument definition does not strictly conform with the semantic role definition by PropBank. For example, "all students" in Example (11) might be A2 (recipient) of predicate 'send' in PropBank.

We define an **event type** as a semantic class of an event mention. Since there is no established general-purpose event ontology that taxonomizes event types, a handful of event types are often predefined to cover and classify event mentions of interest, depending on a particular domain and corpus[4]. For example, the annotation guidelines of the ACE program would determine the event type of E7 in Example (10) as 'Transaction.Transfer-Ownership' where 'Transaction' is an ACE event type and 'Transfer-Ownership' is an ACE event subtype. Some other corpora do not define any event types at all. When we want to represent an event type in example text, we put it under the corresponding event trigger:

(13) John **bought**(E10) a novel at a bookstore yesterday.
E10-A0 Transfer-Ownership E10-A1 E10-LOC E10-TMP

(14) The university will **send emails**(E11) to all students tomorrow.
E11-A0 Phone-Write E11-A1 E11-TMP

(15) The hurricane **left** 20 people **dead**(E12).
E12-A0 E12-A1 Die

[4]See Section 2.1.1 and Section 2.1.2 for examples of the predefined set of event types.

Term	Definition
Event	An abstract representation of a state, activity, accomplishment, achievement, or semelfactive which involves zero or more of participants, attributes, or both.
Event trigger	A main word or phrase in text, typically a verbal or nominal one, which most clearly expresses an event.
Event nugget	A semantically meaningful unit that expresses an event in text.
Event argument	A participant or attribute in text, typically a noun or a noun phrase, which is involved in an event.
Event mention	A clause in text that describes an event, and includes both an event trigger (or nugget) and arguments.
Event type	A semantic class of an event mention.
Realis	The epistemic status of whether an event actually happened or not.

Table 1.1: Definition of terminology regarding events.

One clarification about our definition of events is that an event happens *by its nature*, and whether the event actually happened or not does not matter for event definition. Whether an event actually happened or not is one aspect of events, and we define **realis** as a property of events to represent the epistemic status. For realis status, we use three realis values which are defined in the event nugget annotation task (Mitamura et al., 2015a) for the TAC KBP 2015 Event Track and in the DEFT Rich ERE annotation guidelines (LDC, 2015):

- ACTUAL refers to an event that actually happened at a particular place and time.
- GENERIC refers to a general event that involves types or categories of entities, such as the dying event in "People die."
- OTHER refers to an event that is neither ACTUAL nor GENERIC, such as the meeting event in "He plans to meet with her."

The realis values of E10 and E12 are both ACTUAL since those buying and death events actually occurred. On the other hand, the realis value of E11 is OTHER since it is a future event which has not occurred yet. If we want to further express the realis values in Example (13), Example (14) and Example (15), we show them in the following manner:

(16) John (E13-A0) **bought**(E13) (Transfer-Ownership, ACTUAL) a novel (E13-A1) at a bookstore (E13-LOC) yesterday (E13-TMP).

(17) The university (E14-A0) will **send emails**(E14) (Phone-Write, OTHER) to all students (E14-A1) tomorrow (E14-TMP).

(18) The hurricane (E15-A0) **left** 20 people (E15-A1) **dead**(E15) (Die, ACTUAL).

As a summary of this section, Table 1.1 lists the definitions of terms that we have introduced so far. To give an example of the terms, Table 1.2 provides a slot-filling table representation of E13 in Example (16).

Event		The happening that John bought a novel at a bookstore yesterday
Event trigger		*bought*
Event nugget		*bought*
Event argument	Agent	*John*
	Patient	*a novel*
	Time	*yesterday*
	Location	*a bookstore*
Event mention		*John bought a novel at a bookstore yesterday*
Event type		Transaction.Transfer-Ownership
Realis		ACTUAL

Table 1.2: A slot-filling table representation of the buying event in Example (13).

1.2.4 Other Topics Related to Events

In the previous sections, we described our definition of events, focusing on event spans (nuggets and triggers), arguments, event types, and realis. Besides these properties of events, there are some other important properties of events. Because they are beyond the scope of this thesis, we briefly describe them in this section and leave them for future work.

Duration of Events

The duration of events is concerned with the time at which events take place. More precisely, the duration of events is a period from the time when events start to the time when they end. Our definition of events uses eventualities from the semantic perspective, as described in Section 1.2.3. The notion of eventualities includes both durative events and punctual events, and the time argument of events can specify when they happened. However, in this thesis we do not deal with the duration of events.

One way to formalize the duration of events is interval temporal logic (Allen, 1983; Allen and Ferguson, 1994), which is a specialized form of temporal logic for representing both propositional and first-order logical reasoning about periods of time. Let $X(s, e)$ denote event X where s and e are the start and end time of X, respectively. We also use $=$ and $<$ for temporal expressions to denote that time a equals time b and that a precedes b, respectively. For example, if a is July 19, 2013 and b is July 22, 2013, then $a < b$. Note that $s < e \; \forall X(s, e)$. Interval temporal logic defines 13 possible relations between two events $X(s_x, e_x)$ and $Y(s_y, e_y)$:

- X *equal* Y: $s_x = s_y$ and $e_x = e_y$
- X *before* Y and Y *after* X: $e_x < s_y$
- X *meets* Y and Y *met-by* X: $e_x = s_y$
- X *overlaps* Y and Y *overlapped-by* X: $s_x < s_y < e_x < e_y$
- X *during* Y and Y *contains* X: $s_x < s_y < e_y < e_x$
- X *starts* Y and Y *started-by* X: $s_x = s_y$
- X *finishes* Y and Y *finished-by* X: $e_x = e_y$

Aspects of Events

Aspects of events are another temporal property of events. Aspects are concerned with how events can be situated in relation to a time line. Let us consider the following examples:

(19) She **ate**(E16) a sandwich.

(20) She had **eaten**(E17) a sandwich.

(21) She was **eating**(E18) a sandwich.

Considering the perfect aspect and the progressive aspect, E16 can be viewed as non-progressive and imperfect. On the other hand, E17 are non-progressive and perfect, and E18 are progressive and imperfect. In this thesis, we do not annotate or computationally model them.

More Fine-grained Epistemic Status

Epistemic status of events concerns with whether events actually happened or not. As described in Section 1.2.3, this thesis follows the formalization of realis defined in the TAC KBP Event track and use three realis values: ACTUAL, GENERIC, and OTHER. However, epistemic status of events is deeply connected with the modality system of language, which is more complex. From a perspective of the modality system, the TAC KBP definition of realis can be viewed as one simplification of the modality system that puts various epistemic modalities into the OTHER class. Let us consider the following examples:

(22) She will **eat**(E19) a sandwich.

(23) If she is hungry, she will **eat**(E20) a sandwich.

(24) She needed to **eat**(E21) a sandwich.

(25) She may have **eaten**(E22) a sandwich.

(26) She did not **eat**(E23) a sandwich.

E19 is a future event whereas E22 is a conditioned (hypothetical) event. E21 is necessitated while E22 is speculated. E23 is a negated event. The negation construction through 'not' can also be syntactically applied to the other examples from E19 to E22. Thus, one refined way to deal with the epsitemic status is to employ two notions of polarity and modality, as seen in ACE (LDC, 2005) and Richer Event Description (RED) (Palmer et al., 2016). These kinds of more fine-grained epistemic status are beyond the scope of this thesis.

1.3 Definition of Event Coreference

In this section, we describe our definition of event coreference, which is mainly based on (Hovy et al., 2013). Event coreference is determined by the notion of event identity, and we explain two different types of event identity : full identity (Section 1.3.1) and partial identity (Section 1.3.2).

We define **event coreference** as *a linguistic phenomenon that two event mentions refer to the same event*. Let us consider an example:

(27) John **bought**(E24) a novel at a bookstore yesterday. Mary saw the **purchase**(E25).

We humans know that two event mentions E24 and E25 refer to the same event. Thus, a relation between E24 and E25 is an example of event coreference. However, one question arises: how do we know that two event mentions are identical? In order to define event coreference as the above, we need to define event identity. Since it is difficult to give a clear definition of events as described in Section 1.2, it is also difficult to provide a perspicuous definition of event identity.

1.3.1 Full Identity of Events

We first define the full identity of events as follows. Two event mentions **fully corefer** if the events that they refer to are identical in all aspects, and there is no semantic difference between them. It is possible to replace one mention with the other without any semantic change, although some small syntactic changes might be required to ensure grammaticality. We enumerate several types of the full identity:

1. *Lexical identity*: two mentions use the same sense of the same lexical items, e.g., "move" and "movement".
2. *Pronoun*: one mention refers deictically to the other, e.g., "an earthquake" and "it".
3. *Synonym*: one mention is a synonym of the other, e.g., "wound" and "injure".
4. *Paraphrase*: one mention is a paraphrase of the other. For example, "Mary **gave**(E26) John the book" and "John was **given**(E27) the book by Mary".
5. *Wide-reading*: one mention is a synonym of a wide sense of the other. For example, let us consider the following sentence: "The **attack**(E28) took place yesterday. The **bombing**(E29) killed four people." E28 and E29 are fully coreferent only when "bombing" is read in its wide sense that denotes the whole attack.

Note that the whole attack event expressed by E28 can be referred to by its key event such as "bombing"(E29), and we call such interpretation of "bombing" wide reading. In contrast, "bombing" can refer to a small incident[5] of the whole attack event in some contexts. We call such interpretation narrow reading.

In the ACE program, event coreference was limited to a strict identity of events. Namely, two event mentions were annotated as coreferent if they had the exactly same agent(s), patient(s), time, and location (LDC, 2005). However, there are in practice many coreferent event mentions that violate the strict identity, due to different granularities of event mentions and arguments across documents. Therefore, the Rich ERE (Entities, Relations, Events) standard under the Deep Exploration and Filtering of Text (DEFT) program by DARPA (DARPA, 2012) argue for a more lenient identity of events in event coreference, and propose the notion of **event hopper** (Song et al., 2015). More formally, an event hopper is defined as follows. Two event mentions go into the same event hopper if they meet the following conditions.

- They have the same event type.
- They have the same temporal and location scope, though not necessarily the same temporal expression or specifically the same date, e.g., *attack in Baghdad on Thursday* vs. *bombing in the Green Zone last week*.
- Trigger granularity can be different, e.g., *assaulting 32 people* vs. *wielded a knife*.

[5] We refer to this kind of small event as a subevent. We give a formal definition of subevents in Section 1.3.2.

- Event arguments may be non-coreferential or conflicting, e.g., *18 killed* vs. *dozens killed*.
- Realis status may be different, e.g., *will travel* [OTHER] *to Europe next week* vs. *is on a 5-day trip* [ACTUAL].

To illustrate the more lenient identity of events, let us consider the following example:

(28) There was an **attack**(E30) in Baghdad (E30-LOC) on Thursday (E30-TMP). The **bombing**(E31) in the Green Zone (E31-LOC) last week (E31-TMP) killed 32 people.

E30 and E31 are coreferent because their arguments (location and time) are semantically equivalent, although their surface forms are not exactly the same. This definition of full identity of events is more intuitive and realistic in the sense that it allows for more lenient identities of event arguments. Therefore, we follow the Rich ERE definition of event identity for event coreference in this thesis.

In terms of terminology, event coreference normally refers to a coreferential relation between two event mentions, and thus is a link-based notion. From a clustering perspective, we use the term of an **event coreference cluster** to mean a group of event mentions that are coreferent with each other. By definition, every event coreference cluster contains multiple (coreferential) event mentions, excluding singletons. We also use the term of an **event cluster** to refer to a more general notion meaning either an event coreference cluster or a singleton.

1.3.2 Partial Identity of Events

We now define another type of event identity, which is partial identity of events. Two event mentions **partially corefer** if they refer to the same event, but one mention includes information that is not contained in the other. They are semantically not fully identical, but the core part of the two mentions is. We distinguish between two core types of partial identity: **subevent** and **membership** (Hovy et al., 2013). The essential difference between the two is which aspects of two event mentions in question differ. A membership relation obtains when two event mentions differ in some of the arguments defined in Section 1.2.3, often location and time. In contrast, a subevent relation obtains when two event mentions occur at the more or less same location and time with the same cast of participants, i.e., agent(s) and patient(s).

First, we define a subevent relation as follows: event A is a subevent of event B if B is an event that subsumes a stereotypical sequence of events, or a script (Schank and Abelson, 1977), and A is one of the events. For instance, Example (29) below describes a restaurant script, and E33 is a subevent of E32 because E33 is one of the stereotypical event sequence subsumed by E32.

(29) He had a good **dinner**(E32) last night. He **went**(E33) to a restaurant near his home, and **ordered**(E34) a recommended course menu. He **enjoyed**(E35) beef steak with a glass of red wine.

Since a script represents a stereotypical sequence of events, a relation between two subevent mentions under the same parent carries significant semantic information. Thus, we define such relation as **subevent sister**. A relation between E33 and E34 in Example (29) is an example of subevent sister.

Term	Definition
Event coreference	A linguistic phenomenon that two event mentions refer to the same event.
Subevent	A kind of part-of-whole relation between two event mentions that one represents a stereotypical sequence of events, or a script, and the other is one of events executed as part of that script.
Subevent sister	A relation between two subevent mentions which share the same parent.
Membership	A kind of part-of-whole relation between two event mentions that one represents a set of multiple event instances of the same type, and the other is one or more (but not all) of them.
Event coreference cluster	A group of event mentions that are coreferent with each other.
Event cluster	A group of event mentions that are coreferent with each other or individual singletons.

Table 1.3: Definition of terminology regarding event coreference.

Second, we define a membership relation as follows: event A is a member of event B if B is a set of multiple event instances of the same type, and A is one or more (but not all) of them. Let us consider the following example.

(30) There were three **attacks**(E36) last month. The first **one**(E37) was the most severe.

In this example, E37 is a member of E36 because E37 is one of the specific event instances denoted by E36. Note that E37 is not a subevent of E36 because E36 is not an event that subsumes a certain stereotypical sequence of events. To clarify the different identities of events, we will refer to event coreference based on the full identity of events as **full event coreference**, whereas we refer to event coreference based on the partial identity of events as **partial event coreference**. For brevity, we may refer to full event coreference as event coreference, and partial event coreference as partial coreference. Partial coreference collectively refers to the subevent and membership in this thesis. Table 1.3 summarizes the definition of terms that we have introduced regarding event coreference in this section.

1.3.3 Other Topics Related to Event Coreference

In the previous sections, we defined three event relations: event coreference, subevent, and membership. However, events interact with each other in wider variety of ways, and there are some other important event relations, such as event sequence and causality. Because they are beyond the scope of this thesis, we briefly describe them in this section and leave them for future work.

We give a couple of examples of event sequence and causality:

(31) John **took** a **walk**(E38) after he **finished**(E39) his homework.

(32) Mary **listened**(E40) to music during her whole **drive**(E41) to the office.

(33) The **tsunami**(E42) was caused by the large **earthquake**(E43).

The relation between E38 and E39 is not a subevent or a subevent sister. Rather, it is natural to interpret E38 as a subsequent event of E39. We cannot see stereotypicality in the sequence of E39 and E38. Example (32) shows another temporal relation of events: simultaneity. Example (32) is an event that simultaneously happened with E44. In contrast, Example (33) gives an example of cause-and-effect relationships. In this thesis, we focus on the three event relations of event coreference, subevent, and membership. All the other relaitons such as causality, event sequence and simultaneity are beyond the scope of this thesis, and we leave them for future work.

1.4 Problem Statements

In this section, we describe five important problems with state-of-the-art work on event detection and event coreference resolution: restricted domains in event detection (Section 1.4.1), data sparsity in event detection (Section 1.4.2), lack of subevent detection (Section 1.4.3), event interdependencies via event coreference (Section 1.4.4), and limited applications of events (Section 1.4.5). We address these problems in this thesis.

1.4.1 Restricted Domains in Event Detection

As we will see in Section 3.6, most previous studies on event detection have been conducted in two domains: newswire and biology. Even in these domains, prior work typically focuses on a handful of event types, limiting itself to a particular subset of events. For example, the ACE 2005 corpus defines 33 event types and the TAC KBP 2015 Event Nugget task defines 38 event types (which are similar to those of ACE 2005) in the newswire domain, as described in Section 2.1. In the biology domain, GENIA event extraction is a main task in the BioNLP Shared Task which focuses on events relevant to protein biology, and it defines 9 event types in BioNLP 2009 and 2011 (Kim et al., 2009, 2011) and 13 event types in BioNLP 2013 (Kim et al., 2013).

Most of prior work on event detection explores supervised models trained on such datasets in the newswire and biology domain. Some work explores semi-supervised approaches, such as bootstrapping, to automatically generate additional training data, but these approaches are typically designed specifically for particular domains, and it is unclear how they can scale to other types and domains. Hence, prior work on event detection is restricted in the sense that they are unable to detect events of other types in the respective domains and unable to detect events in other domains. If a new domain X is given for event detection, prior studies would end up repeating the same process as ACE 2005 and GENIA event extraction, developing a domain-specific event corpus for X or developing semi-supervised approaches specifically for X. These kinds of ad-hoc approaches are not a scalable solution for detecting events in various domains, hindering the advance of large-scale event detection.

1.4.2 Data Sparsity in Event Detection

Event detection is the task of identifying event triggers or nuggets in text, and assigning an event type to them. Researchers have employed various structured learning models for event

Model type	System	P	R	F1
Feature-based	JointBeam (Li et al., 2013)	73.7	62.3	67.5
	PatternExpansion (Cao et al., 2015)	68.9	72.0	70.4
	Seed-based (Bronstein et al., 2015)	80.6	67.1	73.2
	JointEventEntity (Yang and Mitchell, 2016)	75.1	63.3	68.7
	PSL (Liu et al., 2016c)	75.3	64.4	69.4
	RBPB (Sha et al., 2016)	70.3	67.5	68.9
Neural network based	CNN (Nguyen and Grishman, 2015)	70.2	65.2	67.6
	DMCNN (Chen et al., 2015)	75.6	63.6	69.1
	JRNN (Nguyen et al., 2016)	66.0	73.0	69.3
	FBRNN (Ghaeini et al., 2016)	66.8	68.0	67.4
	ANN-FN (Liu et al., 2016b)	77.6	65.2	70.7
	HNN (Feng et al., 2016)	84.6	64.9	73.4

Table 1.4: Comparison between reported performances of state-of-the-art systems for event trigger detection on ACE 2005. 'P' and 'R' stand for precision and recall, respectively.

System	Precision	Recall	F1
RPI_BLENDER (Hong et al., 2015)	75.23	47.74	58.41
LCC (Monahan et al., 2015)	73.95	46.61	57.18
LTI (Liu et al., 2015)	73.68	44.94	55.83
UKP (Reimers and Gurevych, 2015)	73.73	44.57	55.56
WIP (Luo et al., 2015)	71.06	43.50	53.97
FBRNN (Ghaeini et al., 2016)	71.58	48.19	57.61

Table 1.5: Comparison between reported performances of state-of-the-art systems for event nugget detection on TAC KBP 2015. The first five systems are the top five official submissions to the TAC KBP 2015 Event Nugget track.

detection, including ones based on structured perceptron with beam search, conditional random fields (CRFs), convolutional neural networks (CNNs), or recurrent neural networks (RNNs)[6]. Despite these efforts, the performance of state-of-the-art event detection models is still far from perfect. This situation is evident from a performance comparison between existing approaches on event trigger detection in Table 1.4 and on event nugget detection in Table 1.5. The comparison indicates that it is difficult for state-of-the-art event detection models to achieve an F1 score of more than 70. This is contrastive to other NLP tasks, such as part-of-speech tagging and named entity recognition, where similar models achieve F1 scores of over 90.

We conjecture that the relatively low performances of the state-of-the-art event detection models shown in Table 1.4 and Table 1.5 arise from some fundamental issues on a human-annotated event corpus. The question is: if there are issues on an event corpus, what are they and why do they prevent a supervised learning model to perform well? There could be several

[6]We provide a literature review of these prior works in Section 3.6.1.

reasons for the task difficulty of event detection, such as inconsistent human annotation. One of important problems which hinders an event detection system from achieving a high performance is *data sparsity* (Li et al., 2014; Chen et al., 2015; Liu et al., 2016b). This problem is common in other NLP tasks where creating human-annotated examples is expensive and time-consuming. In the context of event detection, we state the problem more specifically as follows. An existing human-annotated event corpus is too small for a supervised learning model to achieve decent generalization and capture regularities underlying how event triggers or nuggets of a specific type appear in text[7]. Despite the importance of the problem, no prior work has adequately examined event detection by analyzing the existing event corpora from a data perspective. Methodologies for remedying the data-oriented issues and achieving improvement for supervised learning models have not been well studied in event detection yet.

1.4.3 Lack of Subevent Detection

Subevents themselves are an important fundamental knowledge resource to be extracted from texts. This is because a collection of aggregated subevents (e.g., "go"(E33), "order"(E34) and "enjoy"(E35) in the restaurant script of Example (29)) help to construct a library of domain event backbones (e.g., a family of all stereotypical events under the restaurant script), which can be utilized by other downstream applications. Detecting subevent parent-child relations is also important for full coreference resolution because one can reduce the difficulty of full coreference resolution by excluding subevent relations from candidates of full coreference chains after finding such relations. However, one of the biggest challenges in subevent detection is that some subevent relations exhibit subtle deviation from the full identity of events. This happens because event mentions can refer to events of different semantic granularities. Let us consider the following example:

(34) In the town of Ercis, suspected rebels **fired**(E45) rockets at a police station. No one was injured in the **attack**(E46).

One can say that E46 is a paraphrase of E45, which means that E45 and E46 are fully coreferent. However, one can also argue that E45 is an incident of a larger event E46, which means that E45 is a subevent of E46. To determine the event identity, one must disambiguate the granularity of events from their contexts. This is a challenging problem, and no prior work has studied computational models to detect subevent relations.

There is also no prior work to evaluate computational models for subevent detection, as one can imagine from the lack of such models. Suppose we have built a system that detects subevent relations. It is unclear how to evaluate the performance of the system due to the lack of an evaluation scheme on partial event coreference. There are well-developed algorithms and tools for evaluating full coreference, but they are not readily applicable to partial coreference because unlike full coreference, partial coreference is a directed relation.

Note that membership detection can be also important for the same reason why subevent detection is important for full coreference resolution. However, Hovy et al. (2013) show that the inter-annotator agreement of subevent and membership relations on 65 articles of the Intelligence

[7]We show the smallness of some existing event corpora in Section 2.1.

Community corpus[8] was 0.467 and 0.213, respectively, in terms of Fleiss's kappa. Although the rather low agreement score for the membership coreference is not really reliable given the small size of the corpus, we focus on subevents under the following assumption. That is, we assume that the relatively high inter-annotator agreement for subevent relations implies that one can build a computational model for subevent detection and evaluate it in a meaningful manner.

1.4.4 Error Propagation in Pipeline Models

Events convey semantic information: who did what to whom where and when. They also corefer to each other, playing a role of discourse connection points to form a coherent story. These semantic and discourse aspects of events are not independent of each other, and in fact often work in interactive manners. Let us look at the following examples:

(35) Trebian was **born**(E47) (Be-Born) on November 4th. We were praying that his father would get here on time, but unfortunately he missed **it**(E48) (Be-Born).

(36) In a village near the West Bank town of Qalqiliya, an 11-year-old Palestinian boy was **killed**(E49) (Die) during an exchange of **gunfire**(E50) (Attack). Also Monday, Israeli soldiers **fired**(E51) (Attack) on four diplomatic vehicles in the northern Gaza town of Beit Hanoun, diplomats said. There were no **injuries**(E52) (Injure) from the **incident**(E53) (Attack).

In these examples, E47 corefers to E48, and E50 does to E53. E53 is more abstract than E50, and has less evidence of being a trigger of a specific type. E48 is a pronoun, and thus may seem to refer to an entity rather than an event. State-of-the-art coreference (pronoun) resolvers cannot be helpful to resolve E48 because they are trained for resolving entities. Thus, E48 and E53 are relatively difficult to be recognized as triggers by themselves. However, the event coreferences E47-E48 and E50-E53 help to determine that E48 and E53 are a trigger of 'Be-Born' and 'Attack', respectively. On the other hand, previous works typically rely on a pipelined model that extracts triggers at the first stage, and then resolves event coreference at the second stage. Although this modularity is preferable from development perspectives, the pipelined model limits the interactions. Namely, the first stage alone is unlikely to detect E48 and E53 due to the difficulties described above. These missing events make it impossible for the second stage to resolve the event coreferences E47-E48 and E50-E53.

1.4.5 Limited Applications of Events

There are a wide variety of NLP applications of events, as described in Section 1.1. Nonetheless, there are still unexplored and important areas of applications of events. The existing applications of events can be classified into the following two families.

1. Applications that use semantic (argument) structures of events: automated population of knowledge bases (Ji and Grishman, 2011), question answering (Bikel and Castelli, 2008), text summarization (Li et al., 2006), and stock market prediction (Ding et al., 2014).

[8] See Section 2.1.3 for more details of the corpus.

2. Applications that use event coreference: information extraction (Humphreys et al., 1997), topic detection and tracking (Allan, 2002), textual entailment (Haghighi et al., 2005), and contradiction detection (de Marneffe et al., 2008).

The first family of applications make use of the argument structure of events: who did what to whom where and when. These applications often use events extracted from a given text corpus, assuming that all descriptions of events in the corpus are true. One unexplored area of the applications is a historic perspective of events: whether a particular event mention is historically true or not. This perspective is important especially when people make a complex decision (e.g., buying stock of company X or developing a diplomatic policy against country Y) because they need to analyze relevant events and ensure their belief to support the decision. In the analysis phase, one will probably need to examine which portions of a possibly large amount of textual evidence are trustworthy historical facts. However, despite its importance, the problem of historical true-false judgement of events has not been well studied.

The second family of applications utilize the discourse structure of events via event coreference. Basically, these applications argue that matched pairs of event mentions via event coreference are a useful resource to let a certain system perform a downstream task. However, it is not well studied what impacts event coreferences can have on humans' reading comprehension of texts. In other words, it is unclear how an application can directly involve humans in resolving event coreferences in order to facilitate their semantic understanding of texts. We address these limitations of the existing applications in Chapter 6.

1.4.6 Summary

We have discussed five problems with state-of-the-art work on events and their coreferences. We provide a summary of the problems as follows:

1. *Restricted domains in event detection* (Section 1.4.1): Most prior work on event detection is restricted to closed domains under specific event ontology. Although closed-domain event detection is of practical use in some domain-specific scenarios, it only contributes to partial understanding of events and cannot contribute to advancing natural language applications such as open-domain question answering.
2. *Data sparsity in event detection* (Section 1.4.2): Existing event corpora are relatively small because human annotation of events is normally expensive. This is one of important problems that makes it difficult for supervised learning models to perform event detection adequately as they do in other tasks with a larger amount of labeled data available.
3. *Lack of subevent detection* (Section 1.4.3): Most of prior works on event coreference resolution have focused only on full event coreference. Despite the importance of subevent relations, almost no previous work has explored computational models for subevent detection. Accordingly, there is no prior work to evaluate partial event coreference including subevent relations.
4. *Error propagation in pipeline models* (Section 1.4.4): Previous approaches to event detection and event coreference resolution address these two problems either sequentially or separately, thereby limiting interdependencies of events via event coreference. Errors are cascaded from the event detection phase to the event coreference resolution phase in a

prototypical pipelined approach.

5. *Limited applications of events* (Section 1.4.5): Events have been utilized by numerous NLP applications, but there are still unexplored and important applications of events. Despite its importance, the problem of historical true-false judgement of events has not been well studied. In addition, it is unclear how an application can directly involve humans in resolving event coreferences in order to facilitate human's semantic understanding of texts.

1.5 Goal and Contributions

The central goal of this thesis is to devise a computational method that models the structural property of events in a principled framework for event detection and event coreference resolution. To achieve this goal, we address the five problems described in Section 1.4. The contributions of this thesis are as follows:

1. **Open-domain event detection** (Chapter 3). Most prior work on event detection is restricted to closed domains under specific event ontology, focusing largely on predicate-argument structure. Prior work on open-domain event detection is targeted on limited syntactic types. On the other hand, there is an established consensus that in order to advance natural language applications such as open-domain question answering, we need automatic event identification techniques with larger, wider, and more consistent coverage. To bridge the gap, we propose a new paradigm of *open-domain event detection*, thereby contributing to a wider coverage of events in terms of both domains and syntactic types. The goal is to detect all kinds of events regardless of domains.
2. **Distantly-supervised methods for open-domain event detection** (Chapter 3). We observe that even though closed-domain event detection focuses only on a particular subset of events in particular domains, supervised models cannot generalize well due to the overfitting problem, struggling with small training data. This problem is exaggerated in the open domain because human annotation of events in the open domain is further expensive, due to the ubiquity and ambiguities of events. We show that it is feasible for our distant supervision method to build models detecting events robustly in the open domain while obviating the need for human annotation of events.
3. **Subevent structure detection** (Chapter 4). Our subevent-detection model is the first work to computationally detect subevent parent-child relations as partial event coreference. Paying attention to the fact that subevents form hierarchical event structures, we propose a two-stage approach based on a multiclass logistic regression model, particularly making use of subevent sister relations. We also propose an evaluation scheme for partial event coreference, and show that it is feasible to extend MUC and BLANC while meeting five desiderata of a metric for partial event coreference.
4. **Joint modeling for event detection and event coreference resolution** (Chapter 5). We advocate novel models for event structure detection that jointly detects events and resolves event coreferences in text. Motivated by various observed examples that the semantic and discourse aspects of events often work in interactive manners, we argue that event-oriented computational models that capture the interaction of those aspects will enable

deeper text analysis from both semantic and discourse perspectives, advancing the state-of-the-art technologies for natural language understanding.

5. **Applications of events** (Chapter 6). Most applications of events use event semantics to facilitate natural language understanding by machines. However, capturing event structures is indispensable for natural language understanding by humans as well. Aimed at enhancing the reading comprehension ability of English-as-a-second-language (ESL) students, we present a novel application of event structures for question generation (QG). The application illustrates how the event structures are utilized to overcome limitations of state-of-the-art work. We present an educationally-oriented QG system that generates more sophisticated questions than the traditional QG systems by engaging learners through the use of specific inference steps over multiple sentences.

1.6 Thesis Outline

The rest of this thesis is organized as follows.

- Chapter 2 will describe existing human-annotated corpora of events and their coreferences which we use in this thesis. We will present our evaluation plans to measure the performance of computational models for event detection and full event coreference resolution. In addition, we will propose our evaluation scheme for partial event coreference resolution.
- Chapter 3 will present our approaches to event detection, which comprises two models for detecting event triggers or nuggets, and a method for detecting event arguments using semantic parsing. We will also propose a semi-supervised learning method to automatically expand training data for event detection.
- Chapter 4 will propose a feature-based model and a neural network model for full event coreference resolution. In addition, we will also present our two-stage method for detecting subevent parent-child relations.
- Chapter 5 will propose two document-level structured learning models that jointly extract events and resolve event coreferences. The first employs structured perceptron with joint decoding, and the second one leverages recurrent neural networks.
- Chapter 6 will focus on how events and their coreferences can be utilized in NLP applications. To study the unexplored areas of applications of events, we present our novel approaches to question generation and question answering.
- Lastly, Chapter 7 will give the conclusions of this thesis. It summarizes which pieces of work we have done at the time of the proposal, and which pieces of work we propose with respect to the five problems described in Section 1.4.

Table 1.6 provides a tabular overview of this thesis with respect to the problems described in Section 1.4.

Problem	Assumption to be verified	Relevant part	Evaluation method
Restricted domains in event detection (Section 1.4.1)	It is feasible to achieve high inter-annotator agreement in manual annotation of a wide coverage of events in terms of domains and syntactic types.	Section 2.1.5 and Section 3.2	Averaged pairwise F1 (Section 2.2.1)
Data sparsity in event detection (Section 1.4.2)	Distant supervision allows us to build models detecting events robustly in the open domain while obviating the need for human annotation of events.	Section 3.5	Precision, recall and F1 (Section 2.2.1)
Lack of subevent detection (Section 1.4.3)	A two-stage strategy of finding subevents and selecting their parent is beneficial to detecting subevent parent-child relations.	Section 4.2	BLANC (Section 2.2.2)
	It is feasible to extend MUC and BLANC for evaluating partial event coreference, meeting desiderata substantially well.	Section 2.2.3	Five desiderata for a metric (Section 2.2.3)
Error propagation in pipeline models (Section 1.4.4)	Joint structured learning models are more capable of capturing the event interdependencies, thereby performing event detection and event coreference resolution better than a traditional pipelined model.	Chapter 5	Precision, recall, and F1 (Section 2.2.1) for event detection, and MUC, B^3, CEAF and BLANC (Section 2.2.2) for event coreference resolution
Limited applications of events (Section 1.4.5)	Using event structure, a question generation system can produce more semantically sophisticated questions from multiple sentences than existing ones do from single sentences.	Section 6.1	Grammatical correctness, answer existence and inference steps (Section 6.1.2)

Table 1.6: A structured overview of this thesis with respect to the problems stated in Section 1.4.

Chapter 2

Datasets and Evaluation

In this chapter, we describe human-annotated datasets of events and their coreferences which we use in this thesis (Section 2.1). We then discuss our evaluation schemes to measure the performance of computational models for event detection (Section 2.2.1), full event coreference resolution (Section 2.2.2), and partial event coreference resolution (Section 2.2.3). We describe related work on event corpora and evaluation in Section 2.3. Lastly, we provide a summary of this chapter in Section 2.4. The work described in Section 2.2.3 is based on (Araki et al., 2014a).

2.1 Datasets

In this section, we introduce five event datasets summarized in Table 2.1.

Dataset	Task
ACE 2005 (Section 2.1.1)	Closed-domain event detection (Section 3.4.4)
TAC KBP (Section 2.1.2)	Closed-domain event detection (Section 3.4.4) Event coreference resolution (Section 4.1.4 and Section 5.3.3)
The Intelligence Community corpus (Section 2.1.3)	Subevent detection (Section 4.2.4)
ProcessBank (Section 2.1.4)	Closed-domain event detection (Section 5.2.3) Event coreference resolution (Section 5.2.3) Question generation (Section 6.1)
SW100 (Section 2.1.5)	Open-domain event detection (Section 3.5.4)

Table 2.1: Datasets and associated tasks in which they are used.

2.1.1 ACE 2005

The ACE 2005 corpus[1] (Walker et al., 2006) is a closed-domain event corpus which includes six different document categories: newswire, broadcast news, broadcast conversation, weblog,

[1] https://catalog.ldc.upenn.edu/LDC2006T06

usenet, and conversational telephone speech. Each document in the corpus belongs to one of them. The corpus defines 8 event types and 33 event subtypes, as shown in Table 2.2. In this thesis, we use the same data split as in previous work on event detection, e.g., (Ji and Grishman, 2008), for a comparison with existing methods in the task. This data split has 40 documents in the newswire category for the test set, 30 other documents in different categories for the development set, and 529 remaining documents for the training set. Table 2.3(a) shows statistics of the data split. Prior work on event coreference resolution has not consistently used this data split or others, and different studies use different data splits. This is one of the reasons why it is difficult to compare event coreference resolution systems using the ACE 2005 corpus (Liu et al., 2014). Krause et al. (2016) have recently made their own data split[2] publicly available. We use this data split for event coreference resolution for the sake of a comparison.

Type	Subtype	Type	Subtype	Type	Subtype
Business	Start-Org	Justice	Charge-Indict	Life	Marry
	End-Org		Sue		Divorce
	Declare-Bankruptcy		Convict		Injure
	Merge-Org		Sentence		Die
Conflict	Attack		Fine	Movement	Transport
	Demonstrate		Execute	Personnel	Start-Position
Contact	Meet		Extradite		End-Position
	Phone-Write		Acquit		Nominate
Justice	Arrest-Jail		Appeal		Elect
	Release-Parole		Pardon	Transaction	Transfer-Money
	Trial-Hearing	Life	Be-Born		Transfer-Ownership

Table 2.2: 8 event types and 33 event subtypes defined in the ACE 2005 corpus.

	Train	Dev	Test	Total
# documents	529	30	40	599
# sentences	16473	933	756	18162
# tokens	263740	19091	18760	301591
# triggers	4420	505	424	5349
# arguments	7945	949	899	9793
# clusters	3437	350	303	4090
# singletons	2927	296	240	3463

(a) The data split for event detection.

	Single-word	Multi-word	Subword	All
Verb	2390 (44.7)	187 (3.5)	0 (0.0)	2577 (48.2)
Noun	2460 (46.0)	42 (0.8)	0 (0.0)	2502 (46.8)
Prn	46 (0.9)	0 (0.0)	0 (0.0)	46 (0.9)
Adj	138 (2.6)	0 (0.0)	0 (0.0)	138 (2.6)
Adv	11 (0.2)	2 (0.0)	0 (0.0)	13 (0.2)
Other	36 (0.7)	3 (0.1)	34 (0.6)	73 (1.3)
All	5081 (95.0)	234 (4.4)	34 (0.6)	5349 (100.0)

(b) Event triggers with respect to syntactic types.

Table 2.3: Statistics of the ACE 2005 corpus. In (a), triggers, arguments, and clusters denote event triggers, event arguments, and event (coreference) clusters, respectively. In (b), Prn, Adj, and Adv denote pronoun, adjective, and adverb, respectively. Parentheses show ratios with respect to the percentage.

Table 2.3(b) shows corpus statistics with respect to syntactic types of event triggers. We use Stanford CoreNLP (Manning et al., 2014) to tokenize text, decide head words of event triggers with dependencies, and count multi-word event triggers by part-of-speech tags of their heads.

[2] https://git.io/vwEEP

'Other' in Table 2.3(b) includes demonstrative determiners and particles. As shown, the corpus has a small number of multi-word event triggers, but does not have any discontinuous multi-word ones.

2.1.2 ERE and TAC KBP

The Entities, Relations and Events (ERE) standard was created under the DARPA DEFT program, and initially Light ERE was designed as a lighter-weight version of ACE with the goal of making annotation easier and more consistent (Aguilar et al., 2014). Later, Rich ERE (Song et al., 2015) was created to transition from Light ERE because a richer representation of events within the ERE framework became necessary in the DEFT program. The TAC KBP Event track is a shared task which had been conducted for the three years of 2015[3], 2016[4], and 2017[5]. The task annotates events based on the Rich ERE Annotation guidelines (LDC, 2015) and provides a closed-domain event corpus similar to the ACE 2005 corpus. Both Rich ERE and TAC KBP 2015 define 9 event types and 38 event subtypes (Mitamura et al., 2015a), but TAC KBP 2016 and 2017 reduced the type definition to 7 event types and 18 event subtypes for more efficient dataset creation (Mitamura et al., 2016). Table 2.4 shows the event types. Another main difference from ACE 2005 is that in ERE and TAC KBP, a single event span can be tagged with multiple event types when its event semantics exhibits multiple aspects, corresponding to the types. This issue is called double tagging (Mitamura et al., 2017).[6]

Type	Subtype	Type	Subtype	Type	Subtype
Conflict	Attack	Justice	Arrest-Jail	Personnel	Elect
	Demonstrate	Life	Die		Start-Position
Contact	Meet		Injure		End-Position
	Correspondence	Manufacture	Artifact	Transaction	Transfer-Money
	Broadcast	Movement	Transport-Artifact		Transfer-Ownership
	Contact		Transport-Person		Transaction

Table 2.4: 7 event types and 18 event subtypes defined by the TAC KBP Event track in 2016 and 2017.

Table 2.5 shows the original datasets of Light ERE, Rich ERE and TAC KBP. For consistent comparison with the TAC KBP 2017 dataset, we reduce event nuggets to the 18 event types. We construct a single corpus, which we call the TAC KBP corpus, by combining the ERE and TAC KBP datasets in the following manner. We first set aside the TAC KBP 2017 dataset for the test set. We also put the entire TAC KBP 2014 dataset into the training set because it does not contain gold standard event coreference; we use the TAC KBP 2014 dataset only for training event detection models. We then split the rest of the corpus randomly into training and validation

[3] http://www.nist.gov/tac/2015/KBP/Event/index.html
[4] http://www.nist.gov/tac/2016/KBP/Event/index.html
[5] http://www.nist.gov/tac/2017/KBP/Event/index.html
[6] The name of double tagging seems a little confusing in the sense that it implies two tags. However, there can be single event spans tagged with three or more event types.

Year	# event types	Text type	# docs	# event nuggets	# event clusters	Source
2014	34 (12)	NW	178	4313 (2694)	0 (0)	LDC2014E121 (part
		DF	173	6406 (4089)	0 (0)	of LDC2017E02)
		(Total)	351	10719 (6783)	0 (0)	

(a) Light ERE

Year	# event types	Text type	# docs	# event nuggets	# event clusters	Source
2015	38 (18)	NW	48	1571 (1120)	1099 (793)	LDC2015E29 and
		DF	240	4192 (3693)	3044 (2701)	LDC2015E68
		(Total)	288	5763 (4813)	4143 (3494)	
2016	35 (18)	NW	0	0 (0)	0 (0)	LDC2016E31
		DF	139	3440 (3099)	2680 (2466)	
		(Total)	139	3440 (3099)	2680 (2466)	

(b) Rich ERE

Year	# event types	Text type	# docs	# event nuggets	# event clusters	Source
2015	38 (18)	NW	179	6007 (4684)	3901 (3156)	LDC2015E73 and
		DF	181	6969 (5212)	3559 (2825)	LDC2015E94 (part
		(Total)	360	12976 (9896)	7460 (5981)	of LDC2017E02)
2016	18	NW	85	2505	1862	LDC2016E72 (part
		DF	84	1650	1329	of LDC2017E02)
		(Total)	169	4155	3191	
2017	18	NW	83	2105	1356	LDC2017E54
		DF	84	2270	1607	
		(Total)	167	4375	2963	

(c) TAC KBP

Table 2.5: Statistics of event nugget datasets in LightERE, Rich ERE and TAC KBP. In the text type field, NW and DF refer to newswire and discussion forums, respectively. Parentheses show the numbers when event nuggets are reduced to the 18 event types shown in Table 2.4. Event clusters include both clusters with singletons and ones with multiple coreferential event nuggets, as defined in Section 1.3.1. The Rich ERE 2016 dataset contains discussion forum data only, and each thread is splitted into one or more small units, called CMP; we regard one CMP as one document in the dataset because event coreference is annotated within a CMP unit.

sets, with the ratio of 10:1 in terms of the number of documents. When creating the validation set, we retain the same number of documents from newswire and discussion forums in the set. Table 2.6(a) shows our data split. We have found that 6.7% among all event nugget spans are multi-tagged, which imply the impact of the double tagging problem. Table 2.6(b) shows statistics of the corpus with respect to syntactic types, using the same technique as described in Section 2.1.1. As with ACE 2005, the corpus has a small number of multi-word event nuggets, but does not have any discontinuous multi-word ones. Table 2.6(c) and Table 2.6(d) show event nugget distributions over event types and realis values, respectively.

	Train	Dev	Test	Total
# documents	1187	120	167	1474
# NW documents	430	60	83	573
# DF documents	757	60	84	901
# event nuggets	25964	2782	4375	33121
# event nugget spans	24354	2549	3997	30900
# multi-tagged spans	1520	223	336	2079
# event clusters	13226	1906	2963	18095
# singletons	10394	1477	2358	14229

(a) The data split for event detection.

	Single-word	Multi-word	Subword	All
Verb	17070 (51.5)	1395 (4.2)	0 (0.0)	18456 (55.7)
Noun	12561 (37.9)	340 (1.0)	0 (0.0)	12901 (39.0)
Prn	150 (0.5)	0 (0.0)	0 (0.0)	150 (0.5)
Adj	1012 (3.1)	13 (0.0)	0 (0.0)	1025 (3.1)
Adv	52 (0.2)	4 (0.0)	0 (0.0)	56 (0.2)
Other	212 (0.6)	71 (0.2)	241 (0.7)	524 (1.6)
All	31057 (93.8)	1823 (5.5)	241 (0.7)	33121 (100.0)

(b) Event nuggets with respect to syntactic types.

Event type		Event type	
Conflict.Attack	5238 (15.8)	Manufacture.Artifact	454 (1.4)
Conflict.Demonstrate	1166 (3.5)	Movement.Transport-Artifact	600 (1.8)
Contact.Broadcast	2726 (8.2)	Movement.Transport-Person	3528 (10.7)
Contact.Contact	2214 (6.7)	Personnel.Elect	838 (2.5)
Contact.Correspondence	966 (2.9)	Personnel.End-Position	1513 (4.6)
Contact.Meet	1595 (4.8)	Personnel.Start-Position	676 (2.0)
Justice.Arrest-Jail	1607 (4.9)	Transaction.Transaction	309 (0.9)
Life.Die	3241 (9.8)	Transaction.Transfer-Money	3756 (11.3)
Life.Injure	588 (1.8)	Transaction.Transfer-Ownership	2106 (6.4)

(c) Event nuggets with respect to event types.

Realis	
Actual	18588 (56.1)
Generic	6013 (18.2)
Other	8509 (25.7)

(d) Event nuggets with respect to realis.

Table 2.6: Statistics of the TAC KBP corpus. Parentheses show ratios with respect to the percentage. In (a), NW and DF refer to newswire and discussion forums, respectively. The row of "# multi-tagged spans" shows the number of spans annotated with two or more event types (double tagging). In (b), Prn, Adj, and Adv denote pronoun, adjective, and adverb, respectively. In (d), we found that 11 event nuggets from the TAC KBP 2014 dataset were not annotated with realis (missing gold standard realis values).

2.1.3 Intelligence Community Corpus

The Intelligence Community (IC) corpus consists of 65 newspaper articles in the IC domain. The relations annotated in the corpus can be viewed as the following four classes: full coreference (FC), subevent parent-child (SP), subevent sister (SS), or no coreference (NC). The inter-annotator agreement numbers for FC and SP are 0.620 and 0.467 in terms of Fleiss's kappa, respectively. In addition to relations manually annotated in the corpus, we also consider subevent

relations extended from the combination of FC and SP relations. For instance, if A is a subevent of B, and B is coreferential with C, then A is also a subevent of C. We regarded this type of relation as an SP relation. Table 2.7 shows statistics of the corpus, including our data split. Unlike the ACE 2005 corpus and the TAC KBP corpus, this corpus does not annotate any domain event types, and instead differentiates between domain events and reporting events. We use the corpus for subevent detection.

	Train+Dev	Test	Total
# articles	49	16	65
# relations	26499	9409	35908
FC	1037	216	1253
SP	997	201	1198
SS	399	139	538
NC	24066	8853	32919

Table 2.7: Statistics of the Intelligence Community (IC) corpus.

	Train+Dev	Test	Total
# paragraphs	150	50	200
# event triggers	1047	356	1403
# event arguments	1701	587	2288
# event coreferences	101	30	131
# subevent relations	225	92	317

Table 2.8: Statistics of the ProcessBank corpus.

2.1.4 ProcessBank

The ProcessBank corpus, made available[7] by Berant et al. (2014), consists of 200 paragraphs about biological processes, extracted from the high school level textbook *Biology* (Campbell and Reece, 2005). The corpus includes rich process structures annotated by biologists, shown in Figure 2.1. More specifically, the expert annotation includes entity mentions, entity coreference, event triggers, arguments with semantic roles, and event-event relations such as event coreference and causal relations. Table 2.8 shows statistics of the corpus. Each event in the corpus represents a biological process. A trigger is defined as a word or a phrase that expresses the process most clearly, and an argument is defined as a phrase denoting entities that participate in the process. Unlike the ACE 2005 corpus and the TAC KBP 2015 event nugget corpus, this corpus does not annotate any domain event types.

2.1.5 Simple Wikipedia Corpus (SW100)

SW100 is our human-annotated event corpus in the open domain. Ideally, the corpus should annotate all kinds of events in various domains since its target is unrestricted domains. However, annotating events manually in all domains would be unrealistic. To make the corpus creation manageable while retaining the domain diversity, we select 100 articles in Simple English Wikipedia[8], comprising 10 from each of 10 different domains shown in Table 2.9(a). We set up an initial learning period in which we guide three annotators through our annotation guidelines[9] and answer their questions. They then perform annotation independently using BRAT (Stenetorp et al., 2012). We measure inter-annotator agreement using the pairwise F1 score under two

[7] https://nlp.stanford.edu/software/bioprocess/

[8] https://simple.wikipedia.org

[9] We provide the guidelines in Appendix A.

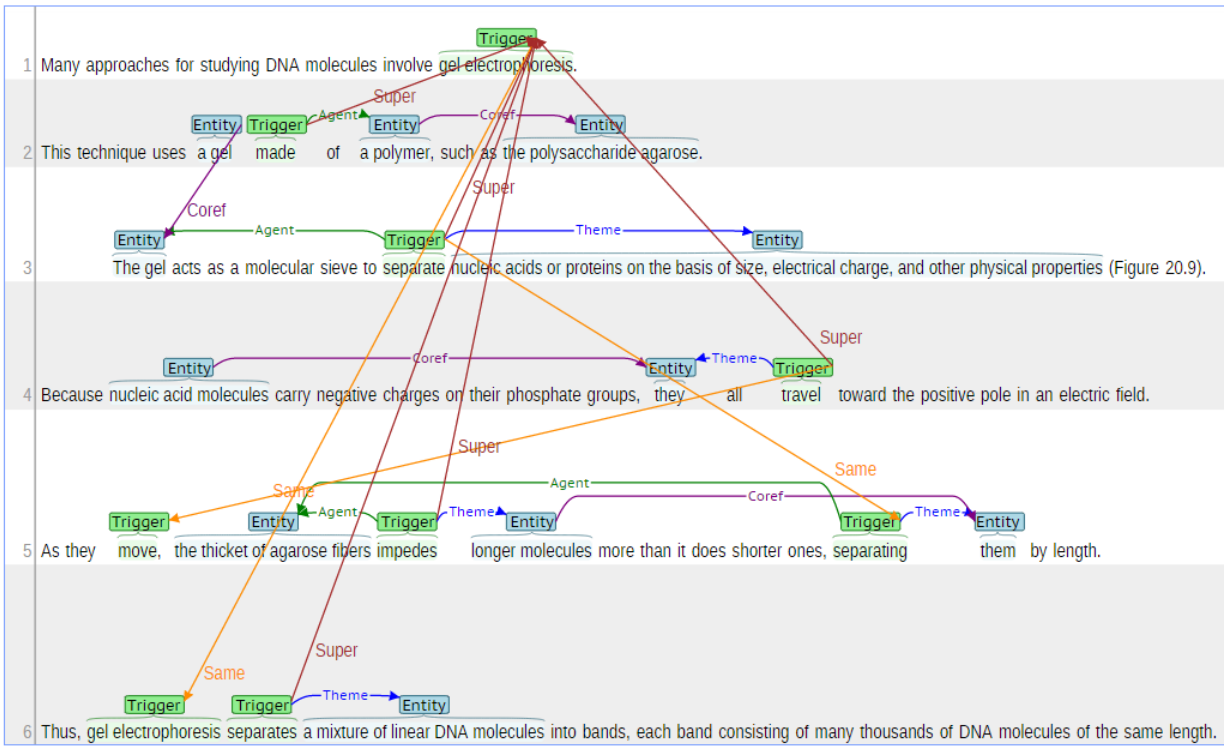


Figure 2.1: An example of expert annotation on a paragraph in ProcessBank, visualized with a modified version of BRAT (Stenetorp et al., 2012). In this example, a purple arrow with 'Coref' represents an entity coreference, an orange one with 'Same' a full (event) coreference, and a red one with 'Super' an opposite of a subevent relation. A green arrow with 'Agent' and a blue one with 'Theme' represent an agent argument and and a theme argument of an event mention, respectively.

conditions: strict match and partial match. The former checks whether two annotations have exactly the same span. The latter checks whether there is an overlap between annotations, with the restriction that each annotation can only be matched to one annotation by the other annotator. We compute the agreements for all annotator pairs and average them for the overall agreement. As a result, the inter-annotator agreement was 80.7% (strict match) and 90.3% (partial match). Finally, the most experienced annotator finalizes event annotation.

Table 2.9(a) shows that event nuggets appear in the 10 domains with comparable frequencies, ensuring the ubiquity of events. We use Stanford CoreNLP (Manning et al., 2014) to tokenize text and decide head words of event nuggets with dependencies. Table 2.9(b) shows corpus statistics of SW100 with respect to syntactic types, using the same technique as described in Section 2.1.1. As shown, multi-word event nuggets amount to 955. 24% of the 955 are discontinuous, and most (97%) of the discontinuous multi-word event nuggets are verb phrases. 'Others' in Table 2.9(b) include pronouns, demonstrative determiners, and numbers.

Domain	# (%)	Domain	# (%)
Architecture	475 (8.8)	Education	653 (12.1)
Chemistry	576 (10.7)	Geology	483 (8.9)
Disaster	510 (9.4)	History	486 (9.0)
Disease	618 (11.4)	Politics	534 (10.0)
Economics	479 (8.9)	Transportation	583 (10.8)

(a) Event nuggets with respect to domains.

	Single-word	Multi-word	All
Verb	2799 (51.9)	560 (10.4)	3359 (62.2)
Noun	1273 (23.6)	382 (7.1)	1655 (30.6)
Adjective	192 (3.6)	2 (0.0)	194 (3.6)
Others	178 (3.3)	11 (0.2)	189 (3.5)
All	4442 (82.3)	955 (17.7)	5397 (100.0)

(b) Event nuggets with respect to syntactic types.

Table 2.9: Corpus statistics of SW100. Percentages (%) are shown in parentheses.

2.2 Evaluation

This section describes how to evaluate computational models for event detection and event coreference resolution. As for event detection (Section 2.2.1) and full event coreference resolution (Section 2.2.2), we follow previous evaluation standards. With respect to partial event coreference (Section 2.2.3), we propose our evaluation method because no evaluation method has been proposed before for this task.

2.2.1 Event Detection

Event detection is analogous to named entity recognition in the sense that their goal is to detect some text spans in text and assign a specific type to them. In this section, we describe how the evaluation of event detection has been done in past studies, paying attention to how it differs from the evaluation schemes of the similar sequence labeling tasks such as named entity recognition. In ACE event extraction, Ji and Grishman (2008) and Ji (2009) define the following evaluation standard to determine the correctness of event detection:

- A trigger is *correctly identified* if its offset (i.e., the position of the trigger word(s) in text) match a reference trigger.

- A trigger is *correctly identified and classified* if its event type and offset match a reference trigger.

We use the latter evaluation scheme for ACE event detection and define precision, recall, and the F1 score as follows:

$$\text{Precision } (P) = \frac{|\text{correctly identified and classified triggers}|}{|\text{all of predicted triggers}|} \tag{2.1}$$

$$\text{Recall } (R) = \frac{|\text{correctly identified and classified triggers}|}{|\text{all of gold standard triggers}|} \tag{2.2}$$

$$\text{F1} = \frac{2PR}{P+R} \tag{2.3}$$

One consideration in this evaluation is whether one should give a system some partial credit if it detects a trigger that partially overlaps a reference trigger. This could happen especially in the cases where a reference trigger is a sub-token, as shown in the following example:

(37) Counter-**demonstrations**(E54) in support of the US-led invasion of Iraq took place in some cities, with some 2,500 people turning out in Chicago.

If a system detects 'Counter-demonstrations' as a trigger in this example, it partially overlaps the reference trigger E54. In this case, the system can be rewarded either no credit or some partial credit. One could avoid the complexities arising from such partial-overlap cases by a strategy that requires every trigger or nugget aligned with token boundaries in the process of creating gold standard data, and performs token-based evaluation.[10] A disadvantage of this strategy is that it makes the annotation process more complicated and expensive. For the purpose of fair comparison with prior work, we use the token-based (without-partial-credit) evaluation for event trigger detection on the ACE 2005 corpus, which is the evaluation standard described above. For the TAC KBP datasets, we follow the character-based (with-partial-credit) evaluation using the event nugget scorer[11] distributed in the TAC KBP Event track (Mitamura et al., 2017). The scorer evaluates performance based on the best mapping between the system output and gold standard under four conditions:

1. span: only spans are considered for the mapping.
2. span+type: types are considered in addition to spans.
3. span+realis: realis is considered in addition to spans.
4. span+type+realis: all attributes are considered (overall).

Following the TAC KBP evaluation standard, we use micro-averaged precision, recall and the F1 score.

On the other hand, for open-domain event detection we consider two matching options: *strict match* and *partial match*. The former checks whether a gold event nugget and a predicted one have exactly the same span. The latter is a relaxed matching option, which checks whether there is an overlap between annotations, with the restriction that each annotation can only be matched to one annotation by the other annotator.

[10] In fact, this evaluation scheme was carried out in the TAC KBP 2015 Event track (Mitamura et al., 2015a).

[11] http://hunterhector.github.io/EvmEval/

Realis Classification

The TAC KBP event nugget scorer described above regards realis as another property of events that we want to predict (in addition to event types) and computes the performance of realis detection using the same evaluation method described above. Additionally, we can think of another subtask, which is *realis classification*. This subtask is to predict the realis value of a given (gold standard) event nugget. Since we define three realis values (see Section 1.2.3), it is a 3-class classification task. We use precision, recall, and F1 to evaluate the performance in each class. We use micro(-averaged) F1 for the overall performance. This metric calculates precision, recall and F1 globally by counting the total true positives, false negatives and false positives. Since the task is a single-label multi-class classification problem, micro F1 is identical to accuracy.

2.2.2 Full Event Coreference Resolution

Recent studies on both entity and event coreference resolution use several metrics to evaluate system performance (Bejan and Harabagiu, 2010; Lee et al., 2012; Durrett et al., 2013; Lassalle and Denis, 2013) since there is no agreement on a single metric. Currently, five metrics are widely used: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), two CEAF metrics $CEAF_e$ and $CEAF_m$ (Luo, 2005), and BLANC (Recasens and Hovy, 2011). These metrics capture different characteristics of coreference output. For an overall comparison with a single metric, we use either of the following two averaged scores: the CoNLL average F1 (Denis and Baldridge, 2009) and the TAC KBP average F1 (Mitamura et al., 2017). These scores are calculated as follows:

$$\text{CoNLL Avg F1} = (\text{MUC F1} + \text{B}^3\text{ F1} + \text{CEAF}_e\text{ F1})/3 \tag{2.4}$$

$$\text{TAC KBP Avg F1} = (\text{MUC F1} + \text{B}^3\text{ F1} + \text{CEAF}_e\text{ F1} + \text{BLANC F1})/4 \tag{2.5}$$

We basically compute these scores using the reference implementation of the entity coreference scorer (Pradhan et al., 2014; Luo et al., 2014).[12] For the TAC KBP dataset, there is an official scorer available,[13] and thus we use the scorer. Note that event coreference evaluation in TAC KBP is different from common entity coreference evaluation with OntoNotes, in the sense that each coreference decision is made at the type level, not at the span level. This implies that the performance of event coreference resolution is heavily influenced by the performance of event type prediction (Mitamura et al., 2017).

2.2.3 Proposed Evaluation for Partial Event Coreference Resolution

In contrast to the well-studied algorithms and tools for evaluating full coreference described in Section 2.2.2, there is no prior work to evaluate computational models for detecting subevent

[12] We use the latest version (v8.01), available at http://conll.github.io/reference-coreference-scorers/. For the TAC KBP average score, we actually use the official scorer of the TAC KBP Event Coreference task (Mitamura et al., 2017) for fair comparison, but this scorer also relies on the reference implementation for event coreference evaluation.

[13] http://hunterhector.github.io/EvmEval/

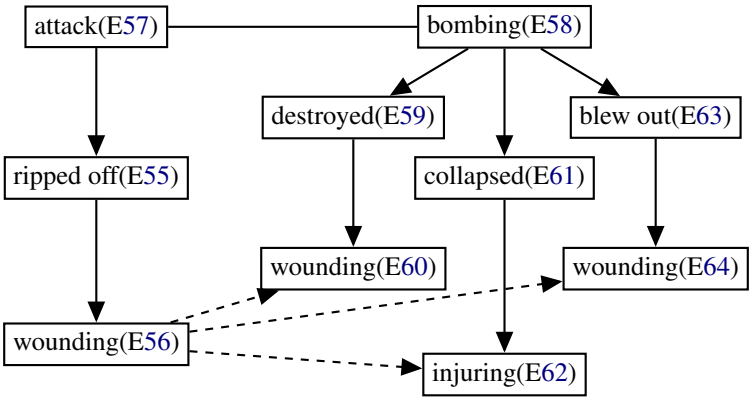


Figure 2.2: Examples of subevent and membership relations. An arrow and a dashed arrow represent a subevent and a membership relation with the direction from a parent to its subevent and member, respectively. For example, E55 is a subevent of E57. Additionally, a straight line represents full coreference.

parent-child relations. This is also the case for the evaluation of membership detection. The algorithms and tools for evaluating full coreference are not readily applicable to partial coreference because unlike full coreference, partial coreference is an directed relation and forms hierarchical event structures. To illustrate this point, Figure 2.2 shows some examples of subevent and membership relations in Example (38).

(38) A car bomb that police said was set by Shining Path guerrillas **ripped off**(E55) the front of a Lima police station before dawn Thursday, **wounding**(E56) 25 people. The **attack**(E57) marked the return to the spotlight of the feared Maoist group, recently overshadowed by a smaller rival band of rebels. The pre-dawn **bombing**(E58) **destroyed**(E59) part of the police station and a municipal office in Lima's industrial suburb of Ate-Vitarte, **wounding**(E60) 8 police officers, one seriously, Interior Minister Cesar Saucedo told reporters. The bomb **collapsed**(E61) the roof of a neighboring hospital, **injuring**(E62) 15, and **blew out**(E63) windows and doors in a public market, **wounding**(E64) two guards.

In this section, we address the problem of evaluating partial coreference resolution, and present an evaluation scheme for partial coreference using a notion of conceptual event heirarchy (Araki et al., 2014a). Designing an evaluation scheme for partial event coreference is important because, as with other tasks, a good evaluation method for partial coreference will facilitate future research on the task in a consistent and comparable manner. When one introduces a certain evaluation metric to such a new complex task as partial event coreference, it is often unclear what metric is suitable to what evaluation scheme for the task under what assumptions. It is also obscure how effectively and readily existing algorithms or tools, if any, can be used in a practical setting of the evaluation. In order to resolve these sub-problems for partial coreference evaluation, we need to formulate an evaluation scheme that defines assumptions to be made regarding the evaluation, specifies some desiderata that an ideal metric should satisfy for the task, and examines how adequately particular metrics can satisfy them. For this purpose, we specifically

investigate three existing algorithms MUC, BLANC, and Simple Tree Matching (STM).

We can divide the metrics for full-coreference into two groups: cluster-based metrics, e.g., B^3 and CEAF, and link-based metrics, e.g., MUC and BLANC. The former group is not applicable to evaluate partial coreference because it is unclear how to define a cluster. The latter is not readily applicable to the evaluation because it is unclear how to penalize incorrect directions of links. We discuss these aspects in Section 2.2.3.

When one formulates an evaluation scheme for a new task, it is important to define assumptions for the evaluation and desiderata that an ideal metric should satisfy. In this section, we first describe assumptions for partial coreference evaluation, and introduce the notion of conceptual event hierarchy to address the challenge posed by one of the assumptions. We then enumerate the desiderata for a metric.

Assumptions on Partial Coreference

We make the following three assumptions to evaluate partial coreference.

Twinless mentions: Twinless mentions (Stoyanov et al., 2009) are the mentions that exist in the gold standard but do not in a system response, or vice versa. In reality, twinless mentions often happen since an end-to-end system might produce them in the process of detecting mentions. The assumption regarding twinless mentions has been investigated in research on entity coreference resolution. Cluster-based metrics such as B^3 and CEAF assume that a system is given true mentions without any twinless mentions in the gold standard, and then resolves full coreference on them. Researchers have made different assumptions about this issue. Early work such as (Ji et al., 2005) and (Bengtson and Roth, 2008) simply ignore such mentions. Rahman and Ng (2009) remove twinless mentions that are singletons in a system response. Cai and Strube (2010) propose two variants of B^3 and CEAF that can deal with twinless mentions in order to make the evaluation of end-to-end coreference resolution system consistent.

In the evaluation of partial coreference where twinless mentions can also exist, we believe that the value of making evaluation consistent and comparable is the most important, and hypothesize that it is possible to effectively create a metric to measure the performance of partial coreference while dealing with twinless mentions. A potential problem of making a single metric handle twinless mentions is that the metric would not be informative enough to show whether a system is good at identifying coreference links but poor at identifying mentions, or vice versa (Recasens and Hovy, 2011). However, our intuition is that the problem is avoidable by showing the performance of mention identification with metrics such as precision, recall, and the F-measure simultaneously with the performance of link identification. In this work, therefore, we assume that a metric for partial coreference should be able to handle twinless mentions.

Intransitivity: As described earlier, partial coreference is a directed relation. We assume that partial coreference is not transitive. To illustrate the intransitivity, let $e_i \xrightarrow{s} e_j$ denote a subevent relation that e_j is a subevent of e_i. In Figure 2.2 on page 31, we have "bombing"(E58) $\xrightarrow{s}$ "destroyed"(E59) and "destroyed"(E59) $\xrightarrow{s}$ "wounding"(E60). In this case, E60 is not a subevent of E58 due to the intransitivity of subevent relations. One could argue that the event "wounding"(E60) is one of stereotypical events triggered by the event "bombing"(E58), and thus E58 $\xrightarrow{s}$ E60. However, if we allow transitivity of partial coreference, then we have to measure all implicit partial coreference links (e.g., the one between E58 and E60) from hierarchical event

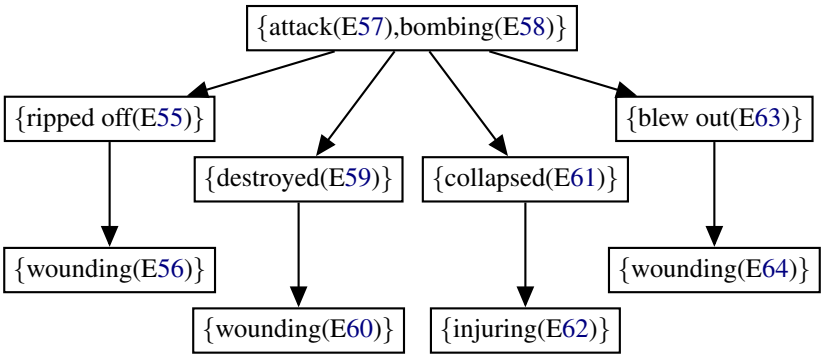


Figure 2.3: A conceptual subevent tree constructed from the full coreference and subevent relations in Figure 2.2.

structures. Consequently, this evaluation policy could result in an unfair scoring scheme biased toward large event hierarchy.

Link propagation: We assume that partial coreference links can be propagated due to a combination of full coreference links with them. To illustrate the phenomenon, let $e_i \Leftrightarrow e_j$ denote full coreference between e_i and e_j. In Figure 2.2, we have "attack"(E57) $\Leftrightarrow$ "bombing"(E58) and "bombing"(E58) $\xrightarrow{s}$ "destroyed"(E59). In this case, E59 is also a subevent of E57, i.e., E57 $\xrightarrow{s}$ E59. The rationale behind this assumption is that if a system identifies E57 $\xrightarrow{s}$ E59 instead of E58 $\xrightarrow{s}$ E59, then there is no reason to argue that the identified subevent relation is incorrect given that E57 $\Leftrightarrow$ E58 and E58 $\xrightarrow{s}$ E59. The discussion here also applies to membership relations.

Conceptual Event Hierarchy

The assumption of link propagation poses a challenge in measuring the performance of partial coreference. We illustrate the challenge with the example in the discussion on link propagation above. We focus only on subevent relations to describe our idea, but one can apply the same discussion to membership relations. Suppose that a system detects a subevent link E58 $\xrightarrow{s}$ E59, but not E57 $\xrightarrow{s}$ E59. Then, is it reasonable to give the system a double reward for two links E58 $\xrightarrow{s}$ E59 and E57 $\xrightarrow{s}$ E59 due to link propagation, or should one require a system to perform such link propagation and detect E58 $\xrightarrow{s}$ E59 as well for the system to achieve the double reward? In the evaluation scheme based on event trees whose nodes represent event mentions, we need to predefine how to deal with link propagation of full and partial coreference in evaluation. In particular, we must pay attention to the potential risk of over-counting partial coreference links due to link propagation.

To address the complexity of link propagation, we introduce a conceptual event tree where each node represents a conceptual event rather than an event mention. Figure 2.3 shows an example of a conceptual subevent tree constructed from full coreference and subevent relations in Figure 2.2 on page 31. Using the set notation, each node of the tree represents an abstract event. For instance, node {E57, E58} represents an attacking event which both event mentions E57 and E58 refer to.

The notion of a conceptual event tree obviates the need to cope with link propagation, thereby

simplifying the evaluation for partial coreference. Given a conceptual event tree, an evaluation metric is basically just required to measure how many links in the tree a system successfully detects. When comparing two conceptual event trees, a link in a tree is identical to one in the other tree if there is at least one event mention shared in parent nodes of those links and at least one shared in child nodes of those links. For example, suppose that system A identifies E57 $\stackrel{s}{\rightarrow}$ E59, system B E58 $\stackrel{s}{\rightarrow}$ E59, system C both, and all the systems identify E57 $\Leftrightarrow$ E58 in Figure 2.2 on page 31. In this case, they gain the same score since the subevent links that they identify correspond to one correct subevent link {E57,E58} $\stackrel{s}{\rightarrow}$ {E59} in Figure 2.3. It is possible to construct the conceptual event hierarchy for membership relations in the same way as described above. This means that the conceptual event hierarchy allows us to show the performance of a system on each type of partial coreference separately, which leads to more informative evaluation output.

One additional note is that the conceptual event tree representing partial coreference is an unordered tree, as illustrated in Figure 2.3. Although we could represent a subevent tree with an ordered tree because of the stereotypical sequence of subevents given by definition, partial coreference is in general represented with a forest of unordered trees[14].

Desiderata for Metrics

In general, a system output of partial event coreference in a document is represented not by a single tree but by a forest, i.e., a set of disjoint trees whose nodes are event mentions that appear in the document. Let T be a tree, and let F be a forest $F = \{T_i\}$. Let $sim(F_g, F_r) \in [0, 1]$ denote a similarity score between the gold standard forest F_g and a system response forest F_r. We define the following properties that an ideal evaluation metric for partial event coreference should satisfy.

P1. *Identity*: $sim(F_1, F_1) = 1$.

P2. *Symmetricity*: $sim(F_1, F_2) = sim(F_2, F_1)$.

P3. *Zero*: $sim(F_1, F_2) = 0$ if F_1 and F_2 are totally different forests.

P4. *Monotonicity*: The metric score should increase from 0 to 1 monotonically as two totally different forests approach the identical one.

P5. *Linearity*: The metric score should increase linearly as each single individual correct piece of information is added to a system response.

The first three properties are relatively intuitive. P4 is important because otherwise a higher score by the metric does not necessarily mean higher quality of partial event coreference output. In P5, a correct piece of information is the addition of one correct link or the deletion of one incorrect link. This property is useful for tracking performance progress over a certain period of time. If the metric score increases nonlinearly, then it is difficult to compare performance progress such as a 0.1 gain last year and a 0.1 gain this year, for example.

In addition, one can think of another property with respect to structural consistency. The motivation for the property is that one might want to give more reward to partial coreference links

[14]For example, it is impossible to intuitively define a sequence of child nodes in a membership event tree in Figure 2.2.

that form hierarchical structures, since they implicitly form sibling relations among child nodes. For instance, suppose that system A detects two links {E57,E58} $\xrightarrow{s}$ {E59} and {E57,E58} $\xrightarrow{s}$ {E61}, and system B two links {E59} $\xrightarrow{s}$ {E60} and {E61} $\xrightarrow{s}$ {E62} in Figure 2.3. We can think that system A performs better since the system successfully detects an implicit subevent sibling relation between {E59} and {E61} as well. Due to space limitations, however, we do not explore the property in this work, and leave it for future work.

Evaluation Metrics

In this section, we examine three evaluation metrics based on MUC, BLANC, and STM respectively under the evaluation scheme described in Section 2.2.3.

B^3 and CEAF. B^3 regards a coreference chain as a set of mentions, and examines the presence and absence of mentions in a system response that are relative to each of their corresponding mentions in the gold standard (Bagga and Baldwin, 1998). Let us call such set a mention cluster. A problem in applying B^3 to partial coreference is that it is difficult to properly form a mention cluster for partial coreference. In Figure 2.3 on page 33, for example, we could form a gold standard cluster containing all nodes in the tree. We could then form a system response cluster, given a certain system output. The problem is that the way B^3 counts mentions overlapped in those clusters cannot capture parent-child relations between the mentions in a cluster. It is also difficult to extend the counting algorithm to incorporate such relations in an intuitive manner. Therefore, we observe that B^3 is not appropriate for evaluating partial coreference.

We see the basically same reason for the inadequacy of CEAF. It also regards a coreference chain as a set of mentions, and measures how many mentions two clusters share using two similarity metrics $\phi_3(R,S) = |R \cap S|$ and $\phi_4(R,S) = \frac{2|R \cap S|}{|R|+|S|}$, given two clusters R and S. One can extend CEAF for partial coreference by selecting the most appropriate tree similarity algorithm for ϕ that works well with the algorithm to compute maximum bipartite matching in CEAF. However, that is another line of work, and due to space limitations we leave it for future work.

Extension to MUC and BLANC. MUC relies on the minimum number of links needed when mapping a system response to the gold standard (Vilain et al., 1995). Given a set of key entities K and a set of response entities R, precision of MUC is defined as the number of common links between entities in K and R divided by the number of links in R, whereas recall of MUC is defined as the number of common links between entities in K and R divided by the number of links in K. After finding a set of mention clusters by resolving full coreference, we can compute the number of correct links by counting all links spanning in those mention clusters that matched the gold standard. It is possible to apply the idea of MUC to the case of partial coreference simply by changing the definition of a correct link. In the partial coreference case, we define a correct link as a link matched with the gold standard one including its direction. Let MUC_p denote such extension to MUC for partial coreference.

Similarly, it is also possible to define an extension to BLANC. Let BLANC_p denote the extension. BLANC computes precision, recall, and F1 scores for both coreference and non-coreference links, and average them for the final score (Recasens and Hovy, 2011). As with MUC_p, BLANC_p defines a correct link as a link matched with the gold standard including its direction. Another difference between BLANC and BLANC_p is the total number of mention

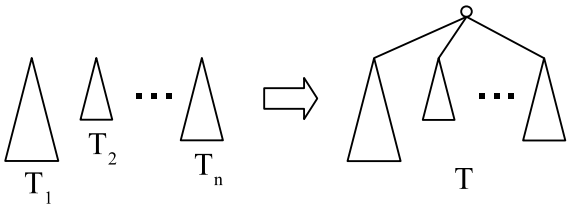


Figure 2.4: Conversion from a forest to a single tree with an additional dummy root.

pairs, denoted as L. In the original BLANC, $L = N(N-1)/2$ where N is the total number of mentions in a document. We use $L_p = N(N-1)$ instead for BLANC$_p$ since we consider two directed links in partial coreference with respect to each undirected link in full coreference.

Extension to Simple Tree Matching. The underlying idea of STM is that if two trees have more node-matching, then they are more similar. The original STM uses a dynamic programming approach to perform recursive node-level matching in a top-down fashion. In the case of partial coreference, we cannot readily use the approach because partial coreference is represented with unordered trees, and thus time complexity of node-matching is the exponential order with respect to the number of child nodes. However, partial event coreference is normally given in a small hierarchy with three levels or less. Taking advantage of this fact and assuming that each event mention is uniquely identified in a tree, we extend STM for the case of unordered trees by using greedy search. Algorithm 1 shows an extension to the STM algorithm for unordered trees.

Algorithm 1 Extended simple tree matching for unordered trees.

Input: two unordered trees A and B
Output: score

```
1:  procedure SimpleTreeMatching(A, B)
2:      if the roots of A and B have different elements then
3:          return 0
4:      else
5:          s := 1                                  ▷ The initial score for the root match.
6:          m := the number of first-level sub-trees of A
7:          n := the number of first-level sub-trees of B
8:          for i = 1 → m do
9:              for j = 1 → n do
10:                 if A_i and B_j have the same element then
11:                     s = s + SimpleTreeMatching(A_i, B_j)
```

We can also naturally extend STM to take forests as input. Figure 2.4 shows how one can convert a forest into a single tree whose subtrees are the trees in the forest by introducing an additional dummy root node on top of those tree. The resulting tree is also an unordered tree, and thus we can apply Algorithm 1 to that tree to measure the similarity of two forests comprising unordered trees. Let STM$_p$ denote the extended STM. Finally, we normalize STM$_p$. Let NSTM$_p$ be a normalized version of STM$_p$ as follows: $NSTM_p(F_1, F_2) = STM_p(F_1, F_2)/max(|F_1|, |F_2|)$ where $|F|$ denotes the number of nodes in F.

Flexibility of Metrics. Making assumptions on evaluation for a particular task and defining

desiderata for a metric determine what evaluation scheme we are going to formulate. However, this kind of effort tends to make resulting evaluation metrics too restrictive to be reusable in other tasks. Such metrics might be adequate for that task, but we also value the flexibility of a metric that can be directly used or be easily extended to other tasks. To investigate the flexibility of MUC_p, BLANC_p and STM_p, we will examine these metrics without making the assumptions of twinless mentions and intransitivity of partial coreference against each metric. We consider that the assumption of link propagation is more fundamental and regard it as a basic premise, and thus we will continue to make that assumption.

MUC was originally designed to deal with response links spanning mentions that even key links do not reach. Thus, it is able to handle twinless mentions. If we do not assume intransitivity of partial coreference, we do not see any difficulty in changing the definition of correct links in MUC_p and making it capture transitive relations. Therefore, MUC_p does not require both assumptions of twinless mentions and intransitivity.

In contrast, BLANC was originally designed to handle true mentions in the gold standard. Since BLANC_p does not make any modifications on this aspect, it cannot deal with twinless mentions either. As for intransitivity, it is possible to easily change the definition of correct and incorrect links in BLANC_p to detect transitive relations. Thus, BLANC_p does not require intransitivity but does require the assumption of no twinless mentions.

Since STM_p simply matches elements in nodes as shown in Algorithm 1, it does not require the assumption of twinless mentions. With respect to intransitivity, we can extend STM_p by adding extra edges from a parent to grandchild nodes or others and applying Algorithm 1 to the modified trees. Hence, it does not require the assumption of intransitivity.

Experiments and Discussions

To empirically examine the three metrics described in Section 2.2.3, we conducted an experiment using the artificial data shown in Table 2.10. Since BLANC_p cannot handle twinless mentions, we removed twinless mentions. We first created the gold standard shown in the first row of the table. It contains fifty events, twenty one singleton events, and seven event trees with three levels or less. We believe this distribution of partial coreference is representative of that of real data. We then created several system responses that are ordered toward two extremes. One extreme is all singletons in which they do not have correct links. The other is a single big tree that merges all event trees including singletons in the gold standard.

We show how the three metrics behave in two cases: (1) we increase the number of correct links from all singletons to the perfect output (equal to the gold standard) in Figure 2.5, and (2) we increase the incorrect links from the perfect output to a single tree merging all trees in the gold standard in Figure 2.6. In the former case, we started with System 3 in Table 2.10. Next we added one correct link $28 \xrightarrow{s} 29$ shown in System 2. This way, we added correct links up to the perfect output one by one in a bottom-up fashion. In the latter case, we started with the perfect output, and then added one incorrect link $49 \xrightarrow{s} 50$ shown in System 1. In a manner similar to case (a), we added incorrect links up to the merged tree one by one in a bottom-up fashion.

The results indicate that MUC_p and BLANC_p meet the desiderata defined in Section 2.2.3 more adequately than NSTM_p. The curve of MUC_p and BLANC_p in both Figure 2.5 and Figure 2.6 are close to the linearity, which is practically useful as a metric. In contrast, NSTM_p fails

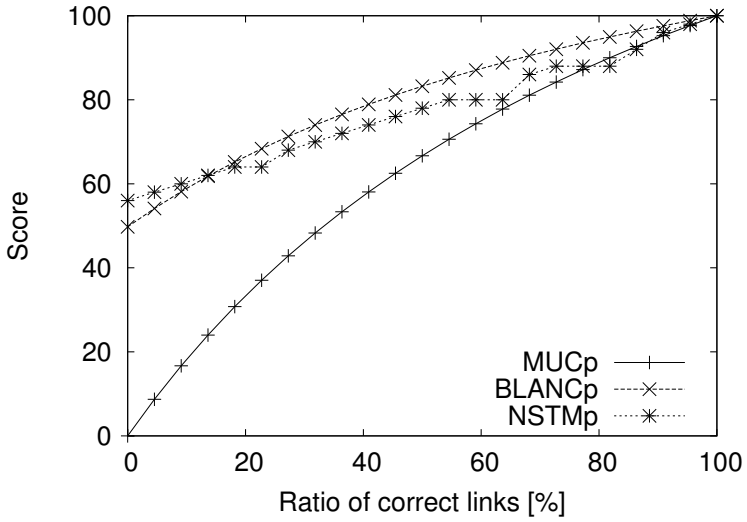


Figure 2.5: Score comparison among MUC_p, BLANC_p, and NSTM_p. The number of correct links increases from singletons to the perfect output (the gold standard) one by one.

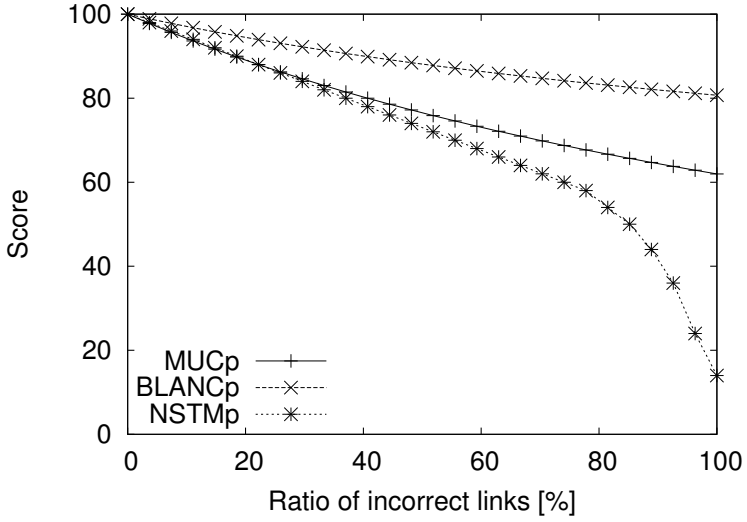


Figure 2.6: Score comparison among MUC_p, BLANC_p, and NSTM_p. The number of incorrect links increases from the perfect output to a single tree merging all trees one by one.

Response	Output
Gold standard	**(1(2(6))(3(7))(4)(5)) (8(9(11)(12))(10)) (13(14)(15)(16)(17)(18)) (19(20(21))(22)) (23(24)(25)) (26(27)) (28(29))** (30) (31) (32) (33) (34) (35) (36) (37) (38) (39) (40) (41) (42) (43) (44) (45) (46) (47) (48) (49) (50)
System 1	**(1(4)(5)(2(6))(3(7))) (8(9(11)(12))(10)) (13(18)(14)(15)(16)(17)) (19(22)(20(21))) (23(24)(25)) (26(27)) (28(29))** (30) (31) (32) (33) (34) (35) (36) (37) (38) (39) (40) (41) (42) (43) (44) (45) (46) (47) (48) **(49(50))**
System 2	(1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20) (21) (22) (23) (24) (25) (26) (27) **(28(29))** (30) (31) (32) (33) (34) (35) (36) (37) (38) (39) (40) (41) (42) (43) (44) (45) (46) (47) (48) (49) (50)
System 3	(1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20) (21) (22) (23) (24) (25) (26) (27) (28) (29) (30) (31) (32) (33) (34) (35) (36) (37) (38) (39) (40) (41) (42) (43) (44) (45) (46) (47) (48) (49) (50)

Table 2.10: Examples of a system response against a gold standard partial coreference. Each event tree is shown in the bold font and in the Newick standard format with parentheses.

to meet P4 and P5 in case (a), and fails to meet P5 in case (b). This is because STM first checks whether root nodes of two trees have the same element, and if the root nodes have different elements, STM stops searching the rest of nodes in the trees.

In Section 2.2.3, we observed that MUC_p and STM_p are more flexible than BLANC_p because they can measure the performance coreference in the case of twinless mentions as well. The experimental results in Section 2.2.3 show that MUC_p and BLANC_p more adequate in terms of the five properties defined in Section 2.2.3. Putting these together, MUC_p seems to be the best metric for partial event coreference. However, MUC has two disadvantages that (1) it prefers systems that have more mentions per entity (event), and (2) it ignores recall for singletons (Pradhan et al., 2011). MUC_p also has these disadvantages. Thus, BLANC_p might be the best choice for partial coreference if we could assume that a system is given true mentions in the gold standard.

Although STM_p fails to satisfy P4 and P5, it has potential power to capture structural properties of partial coreference described in Section 2.2.3. This is because STM's recursive fashion of node-counting can be easily extend to counting the number of correct sibling relations.

2.3 Related Work

In this section, we describe previous works on human annotation of events in Section 2.3.1. The conceptual event hierarchy that we introduced in Section 2.2.3 enables us to evaluate partial event coreference by means of a tree similarity metric. Thus, we review existing work on tree similarity metrics in Section 2.3.2.

2.3.1 Human Annotation of Event Datasets

As described in Section 1.2, events have been defined differently by various researchers. As an obvious result, the ways of annotating events also differ, depending mainly on which definition the annotation is based on. In this section, we describe prior works on human annotation of events.

Closed Domain Events

There is a substantial amount of prior studies on human annotation of events in closed domains. They traditionally focus on limited types of events, mainly defined by several research initiatives and shared tasks in a few domains:

- Newswire: TIPSTER (Onyshkevych et al., 1993), MUC (Grishman and Sundheim, 1996) and STAG (Setzer and Gaizauskas, 2000)
- Multi-domain: ACE (Doddington et al., 2004) and TAC KBP (Mitamura et al., 2016).
- Biology: PASBio (Wattarujeekrit et al., 2004), GENIA (Kim et al., 2008), BioNLP (Kim et al., 2009) and ProcessBank (Berant et al., 2014).

For example, ACE defines 33 types of events such as attacks and elections,[15] and ProcessBank focuses on biological processes.[16] Events in closed domains are of particular interest in some domain-specific scenarios.

Open Domain Events

In contrast to closed-domain events, human annotation of open-domain events is aimed to annotate events in a domain-agnostic manner. OntoNotes (Weischedel et al., 2011) is aimed at covering an unrestricted set of events and entities, but its event annotation is limited to a small number of eventive nouns. TimeML (Pustejovsky et al., 2003) presents a richer specification for annotating events and temporal expressions in natural language text, but does not deal with multi-word and generic events. The annotation guidelines for the ECB+ corpus[17] (Cybulska and Vossen, 2014) present an event-centric annotation task in which only event-related participants, times, and locations were annotated. Our corpus analysis indicates that ECB+ only annotates events in around one third of sentences in a document on average.[18] Mitamura et al. (2015b) propose the idea of event nugget, which is a semantically meaningful unit that expresses the event in a sentence and meets the definition of specific event types and subtypes by itself. As compared to previous human-annotation efforts, this approach allows annotators to tag all possible words that meet the definition of event types and subtypes, including discontinuous phrases, as described in Section 1.2.3. The ERE standards under the DARPA DEFT program first defines Light ERE as a simplified form of ACE annotation with the aim of rapid annotation (Aguilar et al., 2014). They also design Rich ERE (Song et al., 2015), which expands entity, relation and event ontologies, and the notion of taggability. Ritter et al. (2012) address open-domain event detection on

[15] See Section 2.1.1 for details.

[16] See Section 2.1.4 for details.

[17] http://www.newsreader-project.eu/results/data/the-ecb-corpus/

[18] We contacted the authors of ECB+ and confirmed this point with them. It turned out they had not annotated events thoroughly due to annotation cost.

Twitter, and their annotation follows TimeML. Richer Event Description (RED) (Palmer et al., 2016) defines events in a general manner, but its annotation was performed only in the clinical domain (O'Gorman et al., 2016).

Event Coreference

OntoNotes deals with both events and entities, and annotates their coreferences under the same annotation procedure (BBN Technologies, 2006). The human annotation in the ACE 2005 program is limited to a strict identity of events, more specifically, strict match of event arguments. Rich ERE introduces a notion of event hopper, embracing more lenient match of arguments in order to make the annotation of event coreference more realistic and intuitive, as described in Section 1.3. There are some other work on human annotation of event coreference. The EventCorefBank (ECB) corpus by Bejan and Harabagiu (2010) consists of 482 documents from Google News[19] clustered into 43 topics, and annotates within- and cross-document event coreferences. The corpus annotates event expressions by following the TimeML specification (Bejan and Harabagiu, 2008). A disadvantage of the ECB corpus is that it tends to annotate events completely in the first sentence of each document, but not in the rest of the document, which results in the lack of valid event coreference annotations (Liu et al., 2014). Lee et al. (2012) addresses the incomplete event annotation in the ECB corpus through their re-annotation process, providing an extended version[20] of the corpus (extended ECB, or EECB). However, the incomplete-annotation problem still remains in the extended corpus as well. Cybulska and Vossen (2014) further extend the EECB corpus and provide another extension to the ECB corpus, called ECB+, which annotates more intra- and cross-document coreferences.

In relation to partial event coreference, the annotation of bridging references (Haviland and Clark, 1974; Clark, 1977) has been also studied and performed. In a bridging anaphor, an entity introduced in a discourse stands in a particular relation to some previously mentioned discourse entity. The notion of bridging reference has been intensively studied on concrete entities (e.g., (Asher and Lascarides, 1998; Piwek and Krahmer, 2000; Poesio et al., 2004)), but little work has been done on abstract entities such as eventualities. Danlos (2001) discusses particularization and generalization of events as special cases of event coreference. Irmer (2008) represents and resolves bridging relations between events by integrating FrameNet (Baker et al., 1998) and Segmented Discourse Representation Theory (SDRT) (Asher and Lascarides, 2003).

2.3.2 Tree Similarity

By introducing the conceptual event hierarchy described in Section 2.2.3, partial event coreference can be evaluated with a tree similarity metric. We review three existing tree similarity metrics: Tree Edit Distance (TED) and Simple Tree Matching (STM) and tree kernels.

TED is one of the traditional algorithms for measuring tree similarity. It has a long history of theoretical studies (Tai, 1979; Zhang and Shasha, 1989; Klein, 1998; Bille, 2005; Demaine et al., 2009; Pawlik and Augsten, 2011). It is also widely studied in many applications, including Natural Language Processing (NLP) tasks (Mehdad, 2009; Wang and Manning, 2010; Heilman

[19] https://news.google.com
[20] http://nlp.stanford.edu/pubs/jcoref-corpus.zip

and Smith, 2010b; Yao et al., 2013b). However, TED has a disadvantage: we need to predefine appropriate costs for basic tree-edit operations. In addition, an implementation of TED for unordered trees is fairly complex.

STM is another tree similarity metric (Yang, 1991). STM measures the similarity of two trees by counting the maximum match with dynamic programming. Although this algorithm was also originally developed for ordered trees, the underlying idea of the algorithm is simple, making it relatively easy to extend the algorithm for unordered trees.

Tree kernels have been also widely studied and applied to NLP tasks, more specifically, to capture the similarity between parse trees (Collins and Duffy, 2001; Moschitti et al., 2008) or between dependency trees (Croce et al., 2011; Srivastava et al., 2013). This method is based on a supervised learning model with training data; hence we need a number of pairs of trees and associated numeric similarity values between these trees as input. Thus, it is not appropriate for an evaluation setting.

2.4 Summary

In this chapter, we first introduced the datasets that we use for this thesis. We then discussed how to perform the evaluation of event detection and event coreference resolution in this thesis. As for event detection and full event coreference resolution, we reuse the previously developed evaluation standards for the sake of comparison with prior work. However, partial event coreference has no established evaluation scheme. Thus, we proposed an evaluation scheme using conceptual event hierarchy constructed from mention-based event trees. We discussed possible assumptions that one can make, and examined extensions to three existing metrics. Our experimental results indicate that the extensions to MUC and BLANC are more adequate than the extension to STM. To our knowledge, this is the first work to propose an evaluation scheme for partial event coreference. Nevertheless, we believe that our scheme is generic and flexible enough to be applicable to other directed relations of events (e.g., causality and entailment) or other related tasks to compare hierarchical data based on unordered trees (e.g., ontology comparison).

Chapter 3

Event Detection

In this chapter, we focus on the problem of event detection. Based on our definition of events described in Section 1.2.3, we consider two types of *event detection*: closed-domain event detection (Section 3.1) and open-domain event detection (Section 3.2). Event detection is a fundamental step for event coreference resolution (see Chapter 4). In this section, we view event arguments as an important feature for closed-domain event detection and event coreference resolution, and describe how we extract event arguments in Section 3.3. We present our approaches to event detection in Section 3.4 and Section 3.5. We provide a literature review of studies on event detection and event argument detection in Section 3.6. Finally, we summarize this chapter in Section 3.7. The work described in Section 3.5 is based on (Araki and Mitamura, 2018).

3.1 Closed Domain Event Detection

The goal of closed-domain event detection is to identify event triggers or nuggets in text and additionally assign an event type to them. Event types are defined under a particular event ontology for the respective domains. Even in restricted types of closed domains, event detection is a challenging task due to the varieties of event expressions and semantic ambiguities (Li et al., 2013; Nguyen and Grishman, 2015; Araki and Mitamura, 2015). More specifically, various event expressions can exhibit an event of the same type, and the same expression can mean different events of different types, depending on a particular context. For accurate event detection, the task requires not only syntactic analysis but also semantic and discourse analysis of texts. To illustrate the difficulties of the task, we give some examples of event triggers in the ACE 2005 corpus, including some of the examples shown in (Li et al., 2013) and (Li et al., 2014):

(39) A cameraman died when an American tank **fired**(E65) on the Palestine Hotel.
Attack

(40) For this act of stupidity, she was immediately **fired**(E66) from her job.
End-Position

(41) Ellison spent $10.3 billion to **get**(E67) his company.
Merge-Org

(42) We believe that the likelihood of them **using**(E68) those weapons goes up.
Attack

(43) Diller is interested in his own **bid**(E69) for the entertainment unit's assets.
Transfer-Ownership

(44) They had freedom of movement with cars and weapons since the start of the **intifada**(E70).
Attack

Deciding event types of E65 and E66 is difficult because they have the same surface form and are ambiguous in terms of event type assignment. Their event types depend heavily on their contexts. Tagging E67 as 'Merge-Org' and E68 as 'Attack' is also relatively hard because they are common words. These are another kind of examples to suggest that contextual information is important. Another challenge arises from data sparsity. For instance, the same surface forms ('bid' and 'intifada') as E69 and E70 do not appear in the training data of the ACE 2005 corpus.

3.2 Open Domain Event Detection

In Section 1.4.1, we have identified an issue in prior studies on event detection, which is that the domains of event detection are limited. To overcome the issue, we address open-domain event detection. The goal is to detect all kinds of events in text without any specific event types, while not limiting events to any particular domains or types. No event ontology is given in the open domain setting. Below we give several examples of open-domain event nuggets, where we use boldface to highlight event nuggets and underlines to show units of multi-word ones.

(45) The child **broke**(E71) a window of a neighbor's house.

(46) Mary **picked up**(E72) a package in the post office.

(47) Tom **turned** the TV **on**(E73).

(48) The **discussion**(E74) by both groups was ...

(49) By **quality control**(E75) of every step of the production, ...

(50) Total property damage by **Hurricane Katrina**(E76) was around $108 billion.

(51) John was **talkative**(E77) at the party.

(52) She **responded** to his email **dismissively**(E78).

As shown, event nuggets can be either a single word (verb, noun, or adjective) or a phrase which is continuous or discontinuous. For example, E73 in Example (47) is a discontinuous phrasal event nugget, excluding 'the TV'.

3.3 Event Argument Detection with Semantic Parsing

In Section 1.3, we have defined (full) event coreference to be a linguistic phenomenon that the events referred to by two event mentions are identical in all aspects. This definition implies that event argument information plays a crucial role in event coreference resolution. In the context of machine learning, event arguments can be seen as an important feature for event coreference resolution.

For event argument detection, we employ two existing semantic parsers: SEMAFOR[1] (Das

[1] http://www.cs.cmu.edu/~ark/SEMAFOR/

et al., 2014) and the LTH semantic parser.[2] First, SEMAFOR is a frame-semantic parser for English, which extracts semantic frames based on FrameNet from text. It achieves an F1 score of 79.21 for target identification and an F1 score of 68.29 for frame identification (with automatic targets and partial matching) on the SemEval 2007 data set (Das et al., 2014). Second, the LTH semantic parser is a semantic role labeler trained on both PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004). It achieves an unlabeled attachment score (UAS) of 91.17 and a labeled attachment score (LAS) of 85.63 in English on the CoNLL 2009 Shared Task (Björkelund et al., 2009). Both SEMAFOR and the LTH semantic parser are capable of extracting nominal predicates as well as verbal ones. This is crucial for event argument detection because there are a substantial number of nominal event triggers or nuggets. For instance, our corpus analysis shows that approximately 47% of event triggers in the ACE 2005 corpus are nouns, as shown in Table 2.3(b) on page 22.

We regard the output of semantic parsers as partial information of event arguments, and use it in two ways. First, we use it as a feature for event detection since the contextual information such as event arguments are helpful to event type assignment, as illustrated in several examples in Section 3.1. Second, we use it as a feature for event coreference resolution since the definition of event coreference implies that argument information is crucial to event coreference resolution.

3.4 Supervised Closed Domain Event Detection

In our formalization and model design for event detection, we can deal with both event triggers and nuggets in the equivalent manner. Thus, we use only event triggers in the discussion below for brevity. We view event detection as a kind of structured learning problem. More specifically, we regard it as a token-level sequence labeling problem to predict a sequence of event triggers with a type y given input sentence $x = (x_1, ..., x_n)$ where x_i denotes the i-th token in the sentence. In the case of the ACE 2005 corpus with 33 event subtypes,[3] the sequence labeling problem comprises 34-class classification subproblems in each of which one needs to assign to x_i an ACE event subtype or 'None' meaning that x_i is a non-trigger token. In this section, we describe two sequence labeling models: conditional random fields (CRFs) (Section 3.4.1) and bidirectional Long Short-Term Memory (BLSTM) networks (Section 3.4.2). We use the BIO encoding scheme for the both CRF and BLSTM models.

3.4.1 Conditional Random Fields

Conditional random fields (CRFs) (Lafferty et al., 2001) are a popular structured learning model which was traditionally applied to various sequence labeling problems, such as chunking (Sha and Pereira, 2003), POS tagging (Lafferty et al., 2001), and named entity recognition (Finkel et al., 2005). As with such applications, we use the linear-chain CRF that makes a first-order Markov assumption on hidden variables. Given observations X and random variables Y, the

[2] This is a part of the MATE tools (Björkelund et al., 2009), and available at `http://nlp.cs.lth.se/software/semantic-parsing-propbank-nombank-frames/`.

[3] See Section 2.1.1 for more details.

conditional probability distribution defined of the CRF is computed as follows:

$$p_\theta(Y|X) = \frac{\exp(\theta \cdot F(Y, X))}{Z_\theta X} \tag{3.1}$$

where θ denotes feature weights, F a feature function, and

$$Z_\theta = \exp\Big(\sum_y \theta \cdot F(Y, X)\Big) \tag{3.2}$$

The most probable label sequence for input sentence x is:

$$\hat{y} = \arg\max_y p_\theta(y|x) = \arg\max_y \theta \cdot F(y, x) \tag{3.3}$$

The performance of a CRF model relies heavily on underlying features. For the CRF model, we develop features shown in Table 3.1.

Group	Description
Lexical	1. surface form of the current word 2. lemmas and part-of-speech tags of the current, preceding, and following word 3. WordNet synonyms, hypernyms, instance hypernyms, hypernym paths, and topic domains of the first synset associated with the current word and its part-of-speech tag 4. paraphrases of the lemma in PPDB
Syntactic	5. dependency types of head and child dependencies
Semantic	6. name of a lexical unit assigned by SEMAFOR 7. roleset of a predicate assigned by the LTH semantic role labeler 8. names of FrameNet frames associated with the lemma 9. names of parent FrameNet frames of each of the frames found in 8. in terms of relation 'inheritance', 'subframe', or 'perspective_on'

Table 3.1: Features of CRF models for event detection.

3.4.2 Bidirectional Long Short-Term Memory

In this section, we introduce Bidirectional Long Short-Term Memory (BLSTM) networks (Graves and Schmidhuber, 2005). Architecturally speaking, this neural network is within a family of recurrent neural networks, and thus we first describe recurrent neural networks.

Recurrent Neural Networks

Recurrent neural networks (RNNs) (Elman, 1990) is a family of neural networks that models a sequence of vectors to a sequence of hidden states, and has been shown to be a powerful model for sequential data in various fields including natural language processing (NLP). A standard RNN takes as input a word embedding $\mathbf{x}_t$ at time step t, and iteratively computes a hidden state $\mathbf{h}_t$ as follows:

$$\mathbf{h}_t = f(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{3.4}$$

where f is an activation function such as the element-wise logistic sigmoid function. The foundation of the the RNN architecture is a word lookup table, which is essentially a function to convert a given word to its corresponding word embedding. Pre-trained word embeddings are often a preferred choice to initialize the word lookup table. The RNN produces output at the top layer given the hidden state:

$$\mathbf{y}_t = g(\mathbf{h}_t) \tag{3.5}$$

where g is an arbitrary differentiable function. In our model, we apply a linear layer and a softmax layer to produce event type predictions, and a probability of output for each label (event type) l is computed as follows:

$$P(l|x_t) = \text{softmax}_l(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o) \tag{3.6}$$

where $\text{softmax}_l(\mathbf{z}) = \exp(z_l)/\sum_{k=1}^{K}\exp(z_k)$ for $\mathbf{z} = z_1, \ldots, z_K$. Let y_t denote the correct label for input token x_t, and $\hat{y}_t$ denote an output label for x_t:

$$\hat{y}_t = \arg\max_l P(l|x_t) \tag{3.7}$$

Using the categorical cross-entropy loss, the loss function for our RNN-based sequence labeling (event detection) model is computed as follows:

$$L_{seq} = -\sum_t y_t \log(P(\hat{y}_t|x_t)) \tag{3.8}$$

Long Short-Term Memory

In theory, RNNs can model long-range dependencies, but in practice it is difficult due to the problem of gradient vanishing or exploding (Bengio et al., 1994), where gradients might grow or decay exponentially over long sequences. Previous research has explored several variants of RNNs, including Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) networks (Cho et al., 2014). In this thesis, we focus on LSTMs. LSTMs are designed to tackle with the issue by maintaining a separate memory cell $\mathbf{c}_t$ that updates and exposes its content only when necessary. An input gate $\mathbf{i}_t$ controls to what extent the current input passes into the memory cell. A forget gate $\mathbf{f}_t$ controls to what extent the previous memory cell is forgotten. An output gate $\mathbf{o}_t$ controls to what extent the internal memory state is exposed. The hidden state $\mathbf{h}_t$ of an LSTM is computed as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1}) \tag{3.9}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1}) \tag{3.10}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \tag{3.11}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t) \tag{3.12}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{3.13}$$

where σ is the element-wise logistic sigmoid function, and $\odot$ is the element-wise product.

Bidirectional Long Short-Term Memory

As with standard RNNs, one limitation of standard LSTMs is that they process an input sequence unidirectionally, using information only from the past. However, future information can be also useful in event detection. This is because event arguments are often effective features for event detection and some arguments such as patients and locations tend to appear after an event trigger in a sentence. In the case of "fire"(E65) in Example (39) on page 43, for example, "the Palestine Hotel" represents a target facility for the American tank to attack, and this piece of information can also help a model disambiguate E65 to event type 'Attack'. To remedy the issue, we describe another RNN-based neural network in the next section.

Bidirectional Long Short-Term Memory (BLSTM) networks are a variant of LSTMs that enhances LSTMs by modeling a sequence in both forward and backward directions with two separate hidden states to capture past and future information (Graves and Schmidhuber, 2005). BLSTMs have been shown to successfully capture contextual information of a sentence or its subsequence, achieving superior performance in numerous sequence modeling tasks, such as dependency parsing (Wang and Chang, 2016), relation extraction (Miwa and Bansal, 2016), sentiment analysis (Ruder et al., 2016), and question answering (Hermann et al., 2015). Given an input vector $\mathbf{x}_t$ at time step t, let us denote the forward hidden state as $\overrightarrow{\mathbf{h}_t}$ and the backward one as $\overleftarrow{\mathbf{h}_t}$. We obtain the hidden state of a BLSTM using concatenation: $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$.

As with RNNs and LSTMs, BLSTMs can be stacked in layers. The output of the entire architecture is the output of the last layer. It is not theoretically clear why a multi-layered (so called "deep") RNN-based architecture can have more benefits than the single-layered one. However, better performance was empirically observed in some NLP tasks such as named entity recognition (Chiu and Nichols, 2016) and machine translation (Sutskever et al., 2014).

Combining BLSTMs with CRFs

The softmax layer introduced in Equation (3.6) on page 47 produces a probability distribution over labels for each token. However, the classification decision for each token is made independently, and it does not consider correlations between labels in a sequence. On the other hand, the linear-chain conditional random field (CRF) introduced in Section 3.4.1 can be seen as a sequence-level classifier to capture first-order label dependencies. Thus, one can leverage the CRF layer instead of the softmax layer on top of BLSTMs, and this approach is known as **BLSTM-CRF** (Huang et al., 2015). We explore the BLSTM-CRF model because capturing first-order label dependencies with the linear-chain CRF might be useful for event detection as a sequence labeling problem.

Character-level Convolutional Neural Networks

All the neural network models described in the previous section (i.e., RNNs, LSTMs and BLSTMs) initialize input words with their associated word vectors by looking up a word embedding table. For this initialization, using pre-trained word embeddings is common in NLP research. An important benefit of using pre-trained word embeddings is that they are trained on large amounts of text in an unsupervised manner, and supervised models can employ vector representations

for words that do not appear in training data, thereby generalizing better on unseen words. In addition to word embeddings, we also leverage character embeddings through a character-level convolutional neural network (CharCNN) (dos Santos and Zadrozny, 2014) to enhance input representations. CharCNN is a 1-dimensional convolutional neural network over characters, as shown in Figure 3.1. Leveraging character-level representations has two advantages, as shown in previous studies (dos Santos and Zadrozny, 2014; dos Santos and Gatti, 2014; Ma and Hovy, 2016; Chiu and Nichols, 2016). First, it can alleviate the problem of out-of-vocabulary words (words for which we do not have a pre-trained embedding vector) and the inferior quality of word embeddings for rare words. Second, it is an effective approach to capture morphological information such as prefixes and suffixes.

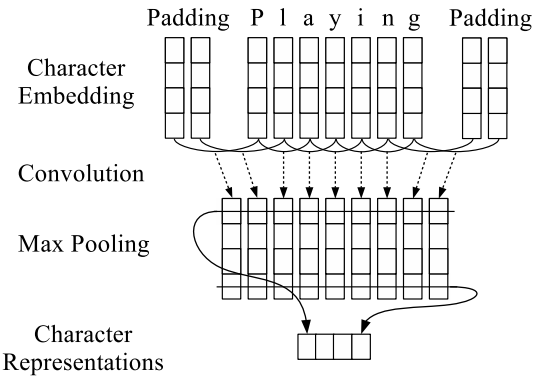


Figure 3.1: A character-level convolutional neural network (CharCNN).

Handling Double Tagging with Multi-label Classification

We have so far addressed event detection as a sequence labeling problem with the BIO encoding scheme, optimizing models to minimize cross-entropy loss. This traditional formalization is well suitable for the setting where each event span is tagged with a single event type (e.g., ACE 2005). However, the formalization is not able to deal adequately with the double tagging problem described in Section 2.1.2. In the TAC KBP setting, handling double tagging is important for both better performance of event detection and that of event coreference resolution. For example, event nugget 'kill' in Example (53) has two event types 'Attack' and 'Die'.

(53) Ronald Reagan ordered airstrikes on Tripoli and Benghazi in April 1986 after an attack on a disco in Germany **killed** (Attack, Die) three people.

To address the issue, we reformalize event detection as a multi-label classification problem where multiple event types can be assigned to a single word or phrase. In this formalization, we do not use the BIO encoding scheme and simply assign one of event types or the 'O' (outside) label. In the inference phase, we regard continuous tokens with the same type as a single event nugget with the type. In the training phase, we optimize the output from the final output layer using multi-label one-versus-all loss based on maximum entropy, instead of the softmax layer

and cross-entropy loss. The loss function for multi-label classification is:

$$L_{mlc} = -\sum_{t}\sum_{i} y_t[i] \log \frac{1}{1+\exp(-\hat{y}_t[i])} + (1-y_t[i]) \log \frac{\exp(-\hat{y}_t[i])}{1+\exp(-\hat{y}_t[i])} \tag{3.14}$$

where i is an index for an event type, $y_t[i]$ is an indicator label of 0 or 1 to specify whether token t has a gold standard event type i, and $\hat{y}_t[i]$ is the output score for token t being assigned to event type i, produced from the final output layer. We refer to the resulting model as **BLSTM-MLC**.

3.4.3 Realis Classification

We have so far discussed our event detection models, which detect event spans with their type. Realis classification is the task of 3-class classification where a system predicts one of three realis values (ACTUAL, GENERIC, and OTHER) for a given event nugget. Thus, this module assumes that event spans are detected beforehand, takes them as input, and predicts their realis value. Figure 3.2 shows a high-level architecture of our realis classification model. The model takes an event nugget as input and operates on event nugget's head word and a sentence including it. We decide the head word by using dependencies from Stanford CoreNLP, as we did in Section 2.1.1. The basic architecture of the realis model is a BLSTM. Additionally, we use CharCNN described in Section 3.4.2 (page 48) and a feedforward neural network with two hidden layers and non-linear activation. We minimize the cross-entropy loss using gold-standard labels.

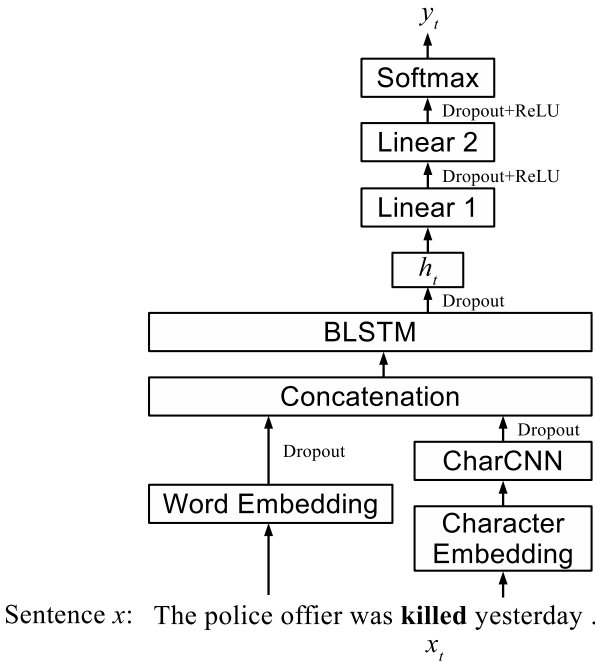


Figure 3.2: A high-level architecture of our realis classification model.

3.4.4 Experiments and Discussions

For our experiments of event detection, we use the ACE 2005 corpus described in Section 2.1.1 and the TAC KBP corpus described in Section 2.1.2. We use the evaluation criteria described in Section 2.2.1 to judge the correctness of predicted event triggers or nuggets.

Results of Closed-domain Event Detection in ACE 2005

In this initial experiment, we compare the performance of the CRF model (Section 3.4.1) and the vanilla LSTM model (Section 3.4.2).

Dataset. We use our own data split of the ACE 2005 corpus for evaluation. We randomly split the corpus into 8:1:1 in terms of the number of documents while keeping the same document distribution over the categories as the original corpus. Table 3.2 shows statistics of the data split.

	Train	Dev	Test	Total
# of documents	479	60	60	599
# of event triggers	4,355	428	566	5,349

Table 3.2: Statistics of our datasets.

Implementation Details. We use the implementation of CRFSuite (Okazaki, 2007) for CRF models. We train the models with L2 regularization using L-BFGS (Liu and Nocedal, 1989). We initially used stochastic gradient descent (SGD) (LeCun et al., 1998), but found that SGD underperformed L-BFGS in our experiments. We tune the regularization parameter for each CRF model on the development dataset by grid search.[4]

As for the LSTM model, we use Adam (Kingma and Ba, 2015) for training because its convergence speed is faster than SGD. We use pre-trained word embeddings from SENNA (Collobert et al., 2011) to initialize the word lookup table. The dimension of the SENNA embeddings is 50. Although SENNA provides embeddings for as many as 130,000 words, we still observed that the ACE 2005 corpus has some words that are not included in the SENNA embeddings. The simplest way to deal with such unknown words is to randomly initialize vectors for them. Instead, we use a technique suggested by Yao et al. (2013a). That is, we choose a small number of words that occur only once in the training dataset, and mark them as `<UNK>`. The learned representation of `<UNK>` through training is used to represent the unknown words in the test dataset. Similarly, we also mark numbers as `<DIGIT>` to learn a single representation for numbers, following Collobert et al. (2011). One hyperparameter of the LSTM model is the dimension of the hidden state. We experiment with six different dimensions of the hidden state $\{50, 100, 200, 300, 400, 500\}$, and compare the performance of the model.

Results. We first apply the CRF model described in (Section 3.4.1) and the LSTM model (Section 3.4.2) to event detection in ACE 2005. Table 3.3 shows the performance of the LSTM model with respect to the six different dimensions of the hidden state. The model achieves the best performance when the hidden state dimension is 300, but the performance difference is quite

[4]Specifically, we selected the best regularization parameter from a set of candidate parameters $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0\}$.

small as compared to the other dimensions of the hidden state. We choose the LSTM model with the 300-dimensional hidden state for the rest of this experiment with ACE 2005.

d_h	Precision	Recall	F1
50	76.18	51.60	61.52
100	71.40	54.43	61.77
200	74.74	51.95	61.30
300	74.51	53.90	62.55
400	73.40	52.84	61.44
500	75.57	53.19	62.43

Table 3.3: The performance of the LSTM model on the test dataset, with respect to different settings of the dimension d_h of the hidden state.

Model	Precision	Recall	F1
CRF	77.57	57.62	66.12
LSTM	74.51	53.90	62.55

Table 3.4: Results of event trigger detection on our test data of the ACE 2005 corpus.

We show the performance of the CRF and LSTM models on our test dataset in Table 3.4. As shown in the table, the CRF model achieved 66.12 F1, outperforming the LSTM model. The reason why the CRF model outperforms the LSTM model seems to be a mixture of the following factors. First, generally speaking, a neural model (e.g., LSTMs) usually requires more data than traditional machine learning models (e.g., CRFs). The ACE corpus is relatively small, and the LSTM cannot generalize well in this particular case. Second, as compared to BLSTMs, vanilla LSTMs model sequences only in a forward direction, lacking future information. Third, the LSTM can have difficulties in capturing long-term dependencies. In contrast, the CRF can directly leverage sequence-level features such as dependencies described in Table 3.1.

Results of Closed-domain Event Detection in TAC KBP

In this experiment, we use the TAC KBP 2017 data[5] and their corresponding official scorer for evaluation, as described in Section 2.2.1. We evaluate the performance of our three event detection models, **BLSTM**, **BLSTM-CRF**, and **BLSTM-MLC** that we presented in Section 3.4.2. BLSTM means a vanilla BLSTM model, BLSTM-CRF stands for the model that combines a BLSTM with a CRF layer, and BLSTM-MLC denotes our BLSTM model to handle double tagging with multi-label classification. We also compare these models with top systems reported in the TAC KBP 2017 task.

Implementation Details. In all the three models, we use the 300-dimensional GloVe vectors trained on a corpus of 42B words[6] from Pennington et al. (2014) and do not fine-tune them during training, thereby ensuring that in-vocabulary words stay close to unseen similar ones for which we have pre-trained vectors. We use one hidden layer with 1000 units. One minibatch corresponds to a sentence, and we set the minibatch size to 32. For optimization, we use Adam (Kingma and Ba, 2015) with the initial learning rate 0.001. In BLSTM-MLC, we also tune a probability threshold as another parameter to cut off type predictions; we perform grid search

[5] See Section 2.1.2 for details.

[6] https://nlp.stanford.edu/projects/glove/.

in the threshold range $\{0.20, 0.21, \ldots, 0.50\}$. For example, if a probability threshold is 0.4, we output type assignments whose probability is larger than 0.4. We train the models for up to 100 epochs, using early stopping based on performance of the span+type F1 score[7] on the validation set.

Results. Table 3.5 and Table 3.6 show precision, recall and the F1 score with respect to spans and types, respectively. 'Top N' in these tables stands for the Nth-ranked system reported in the official results of TAC KBP 2017. Since we optimize our models with respect to type-based loss, the result of Table 3.6 is of more interest.

Model	Precision	Recall	F1
Top 5	58.95	56.53	57.72
Top 4	57.34	61.09	59.16
Top 3	61.74	57.66	59.63
Top 2	64.89	55.71	59.95
Top 1	68.04	**66.53**	**67.27**
BLSTM	**80.85**	47.87	60.13
BLSTM-CRF	80.34	47.03	59.33
BLSTM-MLC	75.34	53.75	62.74

Table 3.5: Performance of event detection with respect to spans.

Model	Precision	Recall	F1
Top 5	57.02	42.29	48.56
Top 4	47.10	50.18	48.60
Top 3	54.27	46.59	50.14
Top 2	52.16	48.71	50.37
Top 1	56.83	**55.57**	56.19
BLSTM	69.79	41.31	51.90
BLSTM-CRF	**70.15**	41.06	51.80
BLSTM-MLC	68.03	48.53	**56.65**

Table 3.6: Performance of event detection with respect to types (span+type).

As shown, our models tend to achieve high precision and outperform the state-of-the-art systems with respect to type prediction. BLSTM-MLC is our best-performing model with respect to span+type prediction. However, our models tend to have relatively low recall. This trend of high precision and low recall is commonly observed in other systems that participated in the task (Mitamura et al., 2017). Since those systems are mostly supervised models trained on TAC KBP data, the trend indicates that supervised models cannot generalize well due to the overfitting problem, struggling with small training data. We also found that the CRF layer is not helpful to event detection. This result is due to the distribution of labels. Multi-word event nuggets (with the 'I-' tags) are only 3.3% among all the event nuggets in the TAC KBP corpus, as shown in Table 2.6b on page 25. Thus, there is little benefit to leverage dependencies between the labels by the linear-chain CRF. The result corresponds to the observation reported by Reimers and Gurevych (2017) in TempEval-3 event detection.

Results of Realis Classification in TAC KBP

In this experiment, we also use the TAC KBP 2017 data to evaluate the performance of our realis classifiers described in Section 3.4.3. We measure the performance using precision, recall and F1, as described in Section 2.2.1.

Implementation Details. For word embeddings, we use the 300-dimensional GloVe vectors trained on a corpus of 42B words and do not fine-tune them during training. We map all out-of-vocabulary words to a zero vector. In CharCNN, we use character embeddings with 15

[7] See Section 2.2.1 for details.

dimensions and 30 filters with window size 3. In BLSTM, we use two hidden layers with 1000 units. The feedforward neural network has two hidden layers with 500 dimensions and rectified linear units (Nair and Hinton, 2010) for non-linear activation. We optimize model parameters using Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001 and a minibatch size of 32. We apply dropout (Srivastava et al., 2014) with 0.5 dropout rate to the word embeddings, the character representations from CharCNN, and hidden layers of BLSTM. We also apply dropout with 0.2 dropout rate to hidden layers of the feedforward neural network. We train the model for up to 100 epochs, using early stopping based on performance on the validation set.

Results. Table 3.7 shows the results. MLP denotes the feedforward neural network (multi-layer perceptron) placed on top of BLSTM, shown in Figure 3.2 on page 50. As shown, incorporating character-level representations through CharCNN improves the performance of realis classification. Table 3.8 shows the confusion matrix of the realis classification results of the BLSTM+CharCNN model. As shown, the classifier makes relatively large error in predicting OTHER when gold standard is ACTUAL.

Model	ACTUAL	GENERIC	OTHER	Overall
BLSTM	83.48	48.87	67.75	73.85
BLSTM+CharCNN	**83.87**	48.94	**68.20**	**74.15**
BLSTM+MLP	82.91	48.40	67.50	73.37
BLSTM+MLP+CharCNN	83.57	**50.23**	66.92	74.10

Table 3.7: Performance of our realis classifiers with respect to the F1 score for each realis value. The F1 score for 'Overall' is computed with micro F1.

		Gold standard		
		A	G	O
Predicted	A	2075	141	191
	G	202	324	191
	O	264	142	845

Table 3.8: The confusion matrix of realis classification by the BLSTM+CharCNN model. 'A', 'G' and 'O' stand for ACTUAL, GENERIC and OTHER, respectively.

To produce end-to-end results of event detection, we form a two-stage pipeline: (1) detection of event spans and types and (2) realis classification. For the second stage, we use the best realis classifier BLSTM+CharCNN. Table 3.9 shows the performance of event detection with respect to realis, and Table 3.10 shows overall performance (end-to-end results). As with Table 3.5 and Table 3.6, 'Top N' in these tables represents the Nth-ranked system reported in the official results. As shown in Table 3.10, our BLSTM-MLC model outperforms the top system of TAC KBP 2017 by 3.92 F1 points in overall evaluation.

Model	Precision	Recall	F1
Top 5	49.86	36.98	42.47
Top 4	43.38	41.60	42.47
Top 3	47.95	46.89	47.42
Top 2	51.39	44.12	47.48
Top 1	46.85	**49.91**	48.33
BLSTM	**64.17**	37.99	47.72
BLSTM-CRF	63.57	37.21	46.95
BLSTM-MLC	58.97	42.07	**49.10**

Table 3.9: Performance of event detection with respect to realis (span+realis).

Model	Precision	Recall	F1
Top 5	35.01	32.70	33.81
Top 4	43.22	32.05	36.81
Top 3	39.69	38.81	39.24
Top 2	42.52	36.50	39.28
Top 1	38.51	**41.03**	39.73
BLSTM	55.09	32.61	40.97
BLSTM-CRF	**55.20**	32.31	40.76
BLSTM-MLC	52.84	37.69	**44.00**

Table 3.10: Overall performance of event detection (span+type+realis).

3.5 Distantly-supervised Open Domain Event Detection

In Section 3.4.4, we observed that even though closed-domain event detection focuses only on a particular subset of events in particular domains, supervised models cannot generalize well due to the overfitting problem, struggling with small training data. The results corroborate the two problems with event detection described in Section 1.4: restricted domains (Section 1.4.1) and data sparsity (Section 1.4.2).

In this section, we describe our distantly supervised approach for open-domain event detection to overcome the problems. Figure 3.3 shows a high-level overview of the approach. As shown, the algorithm comprises two phases: training data generation and event detection. At the core of the approach is distant supervision from WordNet[8] in the former phase to address ambiguities on eventiveness and generate high-quality training data automatically.

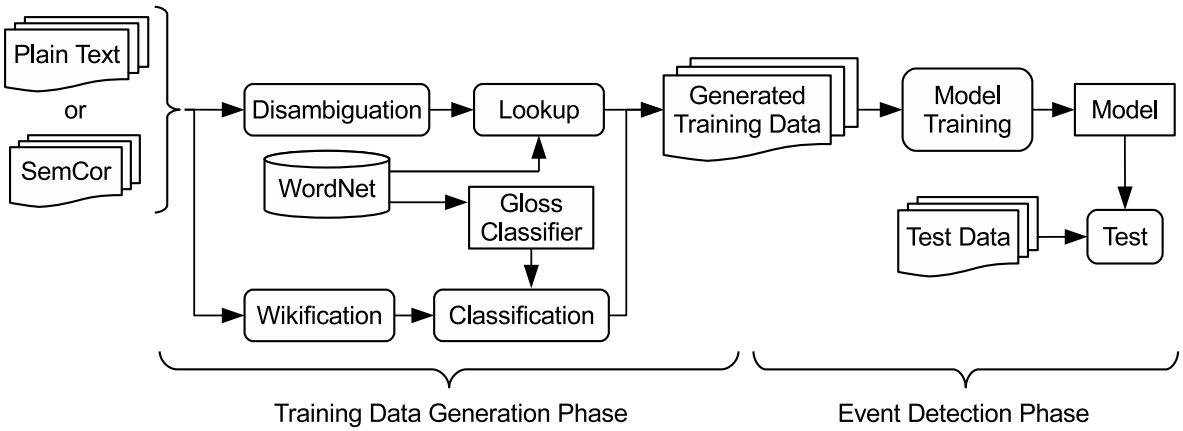


Figure 3.3: An overview of our distantly-supervised open-domain event detection.

[8]We access WordNet 3.0 using NLTK (Bird et al., 2009).

3.5.1 Training Data Generation

The goal of training data generation is to generate high-quality training data automatically from unannotated text. We first disambiguate text using a WordNet-based word sense disambiguation tool. Given sense-annotated text as input, we implement a rule-based algorithm, which we refer to as **RULE**. It is our basis for generating training data. Looking at our annotation guidelines (see Appendix A), we employ several heuristics to identify event nuggets, according to syntactic types:

Verbs. We detect most of main verbs as eventive, excluding be-verbs and auxiliary verbs. However, some exceptions exist. In the examples below, we use italic face to highlight non-eventives.

(54) That is what I **meant**.

(55) 'Enormous' *means* 'very big'.

In Example (54), 'meant' is eventive because it indicates the action of "intend to say" whereas 'means' in Example (55) is not because it merely shows equality, playing the almost same role as a be-verb. Thus, we define a set of non-eventive verb senses and filter out verbs if their disambiguated sense is in the set.

Nouns. There are also ambiguous nouns:

(56) His **payment** was late.

(57) His *payment* was $10.

(58) **Force** equals mass times acceleration.

In Example (56), 'payment' is eventive because it means the action of his paying something while 'payment' in Example (57) is not because it refers to specific money paid by him, which is $10. These examples also show that eventive nouns cannot be simply approximated by verb nominalizations. 'Force' in Example (58) can be easily disambiguated to its physical sense, but still deciding its eventiveness is difficult. To address the issue, we make use of distant supervision from WordNet. Let word_n^i denote the i-th sense (synset) of noun word in WordNet. For example, car_n^1 is the first synset of noun 'car'. Note that eventualities introduced in Section 1.2.1 consists of three components:

- **states**: a class of notions which are durative and changeless, e.g, want, own, love, resemble
- **processes**: a class of notions which are durative and atelic, e.g., walking, sleeping, raining
- **actions**[9]: a class of notions which are telic or momentaneous happenings, e.g., build, walk to Boston, recognize, win, arrive, clap

Looking at textual definitions of synsets called *glosses*, we assume that there is a semantic correspondence between the components and the following WordNet synsets:

- state_n^2: the way something is with respect to its main attributes
- $\mathsf{process}_n^6$: a sustained phenomenon or one marked by gradual changes through a series of states
- event_n^1: something that happens at a given place and time

We detect nouns as events if their disambiguated sense is subsumed by the three synsets above through (instance-)hyponym relations.

Adjectives. Adjectives can also be ambiguous:

[9] Bach (1986) uses term 'events' to refer to this class. In this work, we use 'actions' instead for clarification.

(59) Mary was **talkative** at the **party**.

(60) Mary is a *talkative* person.

In Example (59), 'talkative' is eventive because it implies that Mary talked a lot at the party, whereas 'talkative' in Example (60) is not because it just indicates Mary's personal attribute. As illustrated, major problems with adjectives are to differentiate states from attributes and to figure out if they imply actual occurrences (Palmer et al., 2016). Unlike nouns, no direct supervision is available from WordNet because it does not have any hyponym taxonomies for adjectives. Thus, we use simple and conservative heuristics to detect adjectives as events if they are originated from present and past participles of verbs, illustrated as follows:

(61) There is a **man-made** river in the country.

(62) The tower has 20,000 **sparkling** lights.

Adverbs. We employ a slightly modified version of the heuristics above for adjectives. Adverbs connect with their modifying verbs, forming a single event nugget, as illustrated in Example (52) of page 44. Thus, we combine eventive adverbs with such verbs to detect resulting verb phrases as events.

Phrases. Following Schneider et al. (2014), we define *phrases* to be lexicalized combinations of two or more words that are exceptional enough to be considered as single units in the lexicon. We assume that this definition is suitable to event detection because the exceptionality of multiword units in the phrase lexicon translates to the meaningfulness of textual units of (phrasal) event nuggets. From the perspective of open-domain event detection, supervised phrase detection models are likely suboptimal because they might be limited to particular domains or overfitting to small datasets. Therefore, we explore a simple dictionary-lookup approach to detect WordNet phrases, inspired by Yin and Schütze (2015). One enhancement to their approach is that we examine dependencies using Stanford CoreNLP (Manning et al., 2014), illustrated as follows:

(63) Snipers were **picking** them **off**.

(64) He **picked** an apple off the tree.

In Example (63), 'picking . . . off' forms a discontinuous phrasal verb, whereas 'picked' in Example (64) does not. Dependencies can be of help to resolve these two cases. In the former case, a dependency relation 'picking $\xrightarrow{\texttt{compound:prt}}$ off' is a direct signal of the phrasal verb construction.

3.5.2 Enhancements with Wikipedia

One disadvantage of RULE is the limited coverage of WordNet. In particular, WordNet does not cover many proper nouns that we generally see in newspaper articles, such as the following:

(65) Property damage by **Hurricane Katrina** was around $108 billion.

(66) The **Cultural Revolution** was ...

In order to achieve higher recall, we incorporate Wikipedia knowledge to capture proper nouns which are not in WordNet, motivated by the fact that Wikipedia has a much broader cov-

erage of concepts than WordNet synsets.[10] We use the Illinois Wikifier (Ratinov et al., 2011) to extract Wikipedia concepts from text.

Heuristics-based Enhancement

For our first enhancement, we make two assumptions: (1) the first sentence of a Wikipedia article provides a high-quality gloss of its corresponding concept, and (2) the syntactic head of a gloss represents a high-level concept carrying significant information to decide eventiveness. The first assumption is supported by Wikipedia's style manual on how to write the first sentence of an article.[11] The manual says "If an article's title is a formal or widely accepted name for the subject, display it . . . as early as possible in the first sentence." For instance, the first sentence of entry Electron is:

(67) The electron is a subatomic particle with a negative elementary electric charge.

The gloss of Electron is the underlined text above. Our analysis shows that most Wikipedia articles follow the first-sentence format.

The second assumption is illustrated by the syntactic head of the Electron gloss, which is 'particle'. Based on the assumptions, we develop head-based heuristics, which we call **HeadLookup**. We find the syntactic head of a gloss using dependencies and disambiguate the head using a state-of-the-art word sense disambiguation tool IMS (It Makes Sense) (Zhong and Ng, 2010). We then check if the head's sense is subsumed by the three synsets of state^2_n, $\mathsf{process}^6_n$, and event^1_n. In the case of Electron, the head's sense atom^2_n is not under the synsets. Thus, the model concludes that Electron is non-eventive. Note that HeadLookup itself is a general technique which can be applied to any gloss. Our first enhancement applies it to Wikipedia glosses, and we refer to the enhanced model as **RULE-WP-HL**.

Classifier-based Enhancement

Our second enhancement leverages a binary gloss classifier to decide the eventiveness of proper nouns. We refer to this enhanced model as **RULE-WP-GC**. We use WordNet glosses to train the classifier. Our assumption is that although WordNet and Wikipedia are maintained by different people for different purposes, the classifier trained on WordNet glosses generalizes well against unseen Wikipedia concepts because glosses of the two resources have comparable quality.

Data collection from WordNet. We collect our gloss datasets automatically from WordNet. The goal of data collection is to create a large dataset $D = D_+ \cup D_-$ where D_+ is a set of eventive (positive) glosses, D_- is a set of non-eventive (negative) ones, and $D_+ \cap D_- = \emptyset$. Since WordNet provides a gloss for each synset, the goal reduces to creating a set of positive synsets S_+ and a set of negative ones S_-. Given root synset s, we collect a subset of synsets S_s (including s) by traversing the WordNet taxonomy under s through hyponym and instance-hyponym relations. Using the three synsets introduced in Section 3.5.1, we have $S_+ = S_{\mathsf{event}^1_n} \cup S_{\mathsf{state}^2_n} \cup S_{\mathsf{process}^6_n}$.

[10] WordNet 3.0 has 120K synsets, and English Wikipedia has 5.5M articles as of October 2017, as shown at https://stats.wikimedia.org/EN/TablesWikipediaEN.htm.

[11] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Lead_section#First_sentence

With respect to S_-, we simply take all the WordNet synsets that are not in S_+. Table 3.11 gives several examples of glosses in D_+ and D_-. As shown, the ambiguous word 'payment' has positive and negative synsets. The sizes of D_+ and D_- are $|D_+| = 13,415$ and $|D_-| = 68,700$.

Dataset	Synset	Gloss
D_+	riding^2_n	travel by being carried on horseback
	shower^3_n	a brief period of precipitation
	payment^2_n	the act of paying money
D_-	pork^1_n	meat from a domestic hog or pig
	year^1_n	a period of time containing 365 (or 366) days
	payment^1_n	a sum of money paid or a claim discharged

Table 3.11: Examples of WordNet glosses in D_+ and D_-.

Learning. We train the binary classifier on D using a bidirectional long short-term memory (BLSTM) (Graves and Schmidhuber, 2005). We call the classifier **GC-BLSTM**. Given an input vector $\mathbf{x}_t$ at time step t, let us denote the forward hidden state as $\overrightarrow{\mathbf{h}_t}$ and the backward one as $\overleftarrow{\mathbf{h}_t}$. We obtain the hidden state of a BLSTM using concatenation: $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$. We then use a linear projection of $\mathbf{h}_T$ into two classes: $y = 1$ (eventive) and $y = 0$ (non-eventive). Finally, we add a softmax layer on top of the linear projection, and train the model using the binary cross-entropy loss.

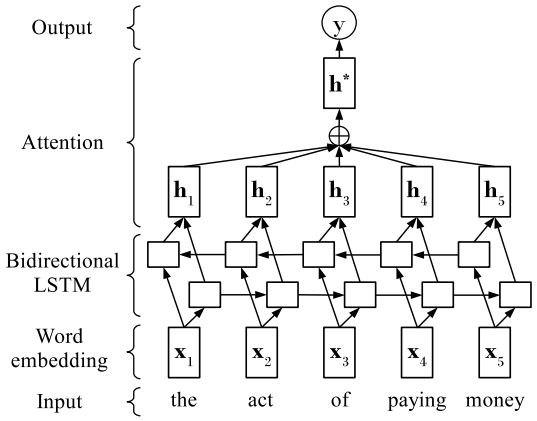


Figure 3.4: The bidirectional LSTM model with a self-attention mechanism.

Attention. Neural networks with attention mechanisms have achieved great success in a wide variety of natural language tasks. The basic idea is to enable the model to attend to all past hidden vectors and put higher weights on important parts so that the model can encode the sequence information more effectively. This idea intuitively makes sense for gloss classification as well, because syntactic heads are likely more important, as illustrated in Section 3.5.2. As shown in Figure 3.4, we leverage a self-attention mechanism, following (Zhou et al., 2016; Lin et al., 2017). Let $\mathbf{H} \in \mathbb{R}^{d \times T}$ denote a matrix comprising hidden vectors $[\mathbf{h}_1, \ldots, \mathbf{h}_T]$ where d is the dimensionality of a hidden vector. The self-attention mechanism computes the hidden state as follows:

$$\mathbf{M} = \tanh(\mathbf{H}) \tag{3.15}$$
$$\alpha = \mathrm{softmax}(\mathbf{w}^T \mathbf{M}) \tag{3.16}$$
$$\mathbf{r} = \mathbf{H}\alpha^T \tag{3.17}$$
$$\mathbf{h}^* = \tanh(\mathbf{r}) \tag{3.18}$$

where a vector $\alpha \in \mathbb{R}^T$ is attention weights and $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector. We refer to the attention-based classifier as **GC-BLSTM-Attn**.

Implementation Details. We use the 300-dimensional GloVe vectors[12] from Pennington et al. (2014) and do not fine-tune them during training. We map all out-of-vocabulary words to a single vector randomly initialized by uniform sampling from $[-0.01, 0.01]$. We use a single hidden layer of 100 dimensions, i.e., $d = 100$. We optimize model parameters using minibatch stochastic gradient descent (SGD) with momentum 0.9. We choose an initial learning rate of $\eta_0 = 1.0 \times 10^{-3}$. We use a minibatch of size 1. To mitigate overfitting, we apply dropout to the inputs and outputs of the network. We also employ L2 regularization. We perform a small grid search over combinations of dropout rates $\{0.0, 0.1, 0.2\}$ and L2 regularization penalties $\{0.0, 1.0 \times 10^{-3}, 1.0 \times 10^{-4}\}$. We use early stopping based on performance on the validation set.

3.5.3 Learning for Event Detection

As seen in the self-training model by Liao and Grishman (2011), erroneously generated training data worsen system performance. Therefore, we need to generate training data as accurately as possible. On the other hand, our algorithm for generating training data comprises at least three non-trivial (error-prone) submodules: disambiguation, wikification, and gloss classification. To eliminate negative effects of disambiguation errors, we choose the SemCor corpus (Miller et al., 1993) as our base text for training data generation. SemCor has human-annotated WordNet senses on 186 documents in numerous genres. We apply our rule-based event detector to generate training data automatically from SemCor.

We formalize event detection as a sequence labeling problem and employ a BLSTM for sequence modeling. One difference from traditional sequence labeling problems is that our output include discontinuous phrases. Thus, we generalize the traditional BIO scheme with two additional tags $\{\mathrm{DB}, \mathrm{DI}\}$. Thus, the BLSTM model computes a hidden representation from each input word and then predicts one of $\{\mathrm{B}, \mathrm{I}, \mathrm{DB}, \mathrm{DI}, \mathrm{O}\}$. Besides the GloVe word embeddings, we use 50-dimensional word embeddings from Turian et al. (2010) and 10-dimensional part-of-speech embeddings. We train the model with the objective of minimizing cross-entropy loss. We use early stopping based on the loss on a validation set.

3.5.4 Experiments and Discussions

In this section, we describe our experimental results and error analysis of eventuality modeling and event detection.

[12]Trained on a corpus of 6B words, they are available at https://nlp.stanford.edu/projects/glove/.

Results of Gloss Classification

Gloss classification is a binary classification subtask in training data generation, aimed to achieve higher recall by capturing proper nouns with Wikipedia knowledge. We randomly sample 1,000 examples from each of the WordNet gloss datasets D_+ and D_- to create a test set and a validation set, and use the rest of D for a training set. We first evaluate our gloss classifiers using the test set. However, what we actually care about is the performance of gloss classification against Wikipedia concepts, as motivated at the beginning of Section 3.5.2. Therefore, we create an additional dataset comprising Wikipedia concepts that do not appear in WordNet. We collect 100 eventive and 100 non-eventive Wikipedia glosses in 10 domains[13], independently of SW100. We measure the performance of gloss classification using accuracy. Table 3.12 shows that GC-BLSTM-Attn performs best and is significantly better than GC-BLSTM on both WordNet and Wikipedia datasets. **BoW-LR** is a bag-of-words model trained with logistic regression, and **DAN** is a deep average network proposed by Iyyer et al. (2015). DAN can be seen as a neural bag-of-words model, and these two models are word-order insensitive baselines. As shown in Table 3.12, the BLSTM model with the attention mechanism has achieved over 91% on the WordNet dataset and 85% on the Wikipedia dataset. Give that the random guess would get 50% accuracy, this result verifies our assumptions that the classifier trained on WordNet glosses can achieve reasonably good performance against unseen Wikipedia concepts and that the attention mechanism is effective for gloss classification.

Model	WordNet	Wikipedia
HeadLookup	77.80	73.50
BoW-LR	79.50	73.00
DAN	83.15	64.00
GC-BLSTM	90.10	80.00
GC-BLSTM-Attn	**91.65****	**85.00***

Table 3.12: Accuracy of gloss classifiers on the datasets from WordNet and Wikipedia. The stars indicate statistical significance compared to the GC-BLSTM model (*: $p < 0.05$; **: $p < 0.005$) based on McNemar's test.

Model	Strict			Partial		
	P	R	F1	P	R	F1
VERB (Baseline)	79.5	51.7	62.7	95.4	62.0	75.2
PRED (Baseline)	55.1	62.4	58.5	67.6	76.6	71.8
RULE	80.1	77.0	78.5	89.0	85.5	87.2
RULE-WP-HL	80.5	77.5	79.0	88.6	85.3	86.9
RULE-WP-GC	80.8	77.7	**79.2**	89.1	85.7	**87.3**

Table 3.13: Performance of the rule-based event detectors on SW100.

Results of Training Data Generation

We measure the performance of the rule-based event detectors on SW100 using precision (P), recall (R), and F1 with the two matching options (strict match and partial match) described in Section 2.2.1. We use IMS (It Makes Sense) for disambiguation. Table 3.13 shows the results. We compare with two baselines.[14] **VERB** is a simple baseline that detects all single-word main verbs as events, excluding be-verbs and auxiliary verbs. **PRED** is another baseline that detects all

[13]Economics, history, politics, psychology, architecture, earth science, physics, chemistry, biology, and medicine.

[14]We also tried Caevo (Chambers et al., 2014) and ClearTK (Bethard et al., 2014) for comparison. However, their performance was quite low due to different nature of the task.

predicates as events by running a state-of-the-art semantic role labeler called PathLSTM[15] (Roth and Lapata, 2016). Since PathLSTM is trained on both PropBank and NomBank, it is able to detect both verbal and nominal predicates. However, semantic role labeling has a different focus on predicate-argument structures. More specifically, the combination of PropBank and NomBank have a narrower coverage of events while having non-event predicates. This difference explains the relatively low performance of 58.5 strict F1, even underperforming the VERB baseline. The performance difference between RULE and VERB mostly comes from nouns, indicating that our WordNet-based heuristics is effective. We found that RULE-WP-GC achieves the best F1 score. This result shows that the proposed enhancements with Wikipedia can contribute to generating higher-quality training data.

We then applied the best-performing RULE-WP-GC to SemCor and found that the generated data contains 59,796 event nuggets in total. We randomly split this data or its subset to 9:1 with respect to the number of documents, creating training and validation data. We train the neural event detector described in Section 3.5.3 on the training data and measure its performance on SW100. Figure 3.5 shows how the amount of training data affects the performance of event detection. As shown, a larger amount of training data enables the model to achieve better performance.

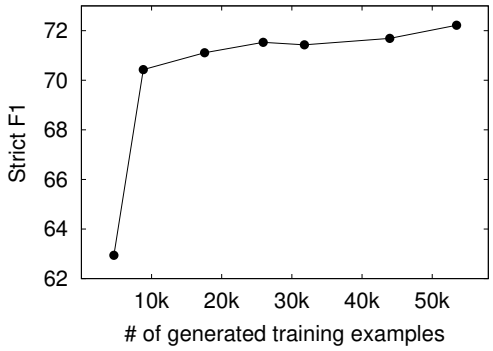


Figure 3.5: Performance of the event detection model on SW100 with respect to the number of training examples generated from SemCor.

Setting	Model	Strict F1	Partial F1
In-domain	BLSTM	73.8	85.9
	DS-BLSTM	**76.1**	**88.0**
Out-domain	BLSTM	67.9	82.8
	DS-BLSTM	**71.3**	**86.6**

Table 3.14: Results of event detection.

Comparison with Supervised Models

In order to test the robustness of our distant supervision model, we compare the model with supervised models in this experiment. We divide SW100 into in-domain and out-domain datasets by randomly sampling 5 domains for each. We further split the in-domain dataset into training (60%), validation (20%) and test subsets (20%) with respect to the number of documents, and then train the BLSTM event detection model described in Section 3.5.3. We repeat this procedure three times and measure the average of F1 scores with strict match (Table 3.14). We refer to the supervised model trained on the in-domain data as **BLSTM** and the distantly supervised model

[15] https://github.com/microth/PathLSTM

trained on the generated data as **DS-BLSTM**. In all the three runs, DS-BLSTM outperforms BLSTM in both in-domain and out-domain settings. The performance difference in the out-domain setting is statistically significant at $p < 0.05$, based on a two-tailed paired t-test. BLSTM ends up overfitting in the in-domain setting, and its weak generalization power is more evident in the out-domain setting. In contrast, DS-BLSTM performs robustly in both settings. Table 3.15 shows the performance of DS-BLSTM in each of the 10 domains. This result indicates that the distantly supervised model performs robustly in various domains.

Domain	Strict			Partial		
	P	R	F1	P	R	F1
Architecture	81.1	71.2	75.8	92.3	81.1	86.3
Chemistry	75.5	71.2	73.3	91.0	85.8	88.3
Disaster	80.7	70.6	75.3	95.7	83.7	89.3
Disease	66.6	53.2	59.2	89.9	71.8	79.9
Economics	73.7	67.8	70.7	93.2	85.8	89.3
Education	71.8	67.4	69.5	86.9	81.6	84.2
Geology	78.6	71.6	75.0	92.3	84.1	88.0
History	77.4	69.8	73.4	92.2	83.1	87.4
Politics	78.2	69.7	73.7	93.5	83.3	88.1
Transportation	81.5	75.6	78.5	91.5	84.9	88.1

(a) With respect to domains.

Syntactic type	Strict			Partial		
	P	R	F1	P	R	F1
Verbs	79.7	83.3	81.5	94.9	99.2	97.0
Nouns	67.4	51.7	58.5	82.5	63.4	71.7
Adjectives	68.9	26.3	38.1	68.9	26.3	38.1

(b) With respect to syntactic types.

Table 3.15: Detailed performance of DS-BLSTM.

Analysis of Gloss Classification

One error pattern is that the gloss of an eventive example is partially or completely overlapped with that of non-eventive one, confusing the classifier:

- broadcast^2_n: a radio or television show
- laugh_track^1_n: prerecorded laughter added to the soundtrack of a radio or television show

The former is eventive and the latter is not. Beside such errors, we found possible inconsistencies in WordNet entries in terms of eventiveness:

- sufficiency^1_n: sufficient resources to provide comfort and meet obligations
- unanimity^1_n: everyone being of one mind
- minority^3_n: any age prior to the legal age

The first three synsets are in the state^2_n taxonomy, but the ways of defining them, especially the syntactic heads of glosses (underlined above) sound like non-eventive entities rather than events.

Analysis of Training Data Generation

Besides gloss classification, our training data generation is subject to errors from the rule-based event detection and wikification. Table 3.16 shows noticeable errors in training data generation. The cause of error (1) is small coverage of WordNet phrases, particularly verb phrases. Error (2) and (4) can be reduced by more sophisticated wikification and gloss classification, respectively. An example of error (3) is that Wikipedia entry Spanish_flu is empty because it is redirected

#	Submodule	Description	Examples
(1)	Phrase detection	A phrase is missing or incorrectly detected.	amount to, stay clear, take one's toll
(2)	Wikification	A proper name is not identified or disambiguated into an incorrect entry in Wikipedia.	Polish Revolution, Battle of The Little Horn
(3)	Wikipedia gloss extraction	A corresponding Wikipedia article does not provide a gloss of an expected form.	Spanish flu, Exxon Valdez oil spill
(4)	Gloss classification	A disambiguated gloss is misclassified.	Anglican parson, Archbishop

Table 3.16: Noticeable errors of our training data generation.

to 1918_flu_pandemic. A simple remedy of partial help to error (3) is to resolve such empty concepts using Wikipedia redirect relations.

Analysis of Event Detection

Two major error sources of the DS-BLSTM model are nouns and phrases. Our training data generated from SemCor is 11 times larger than SW100 with respect to the number of event nuggets. Still, many nouns and phrases do not appear in the training data, making correct predictions difficult. As shown in Table 3.15(a), the most difficult domain is 'Disease' where numerous domain-specific terms, such as migraine and bubonic plague, can appear even in simplified text of Simple Wikipedia, but not in SemCor at all.

3.6 Related Work

In this section, we provide a literature review of event detection (Section 3.6.1) and event argument detection (Section 3.6.2). We also describe prior work on semi-supervised learning approaches in NLP (Section 3.6.3).

3.6.1 Event Detection

As mentioned in Section 1.4.1, event detection has been studied mainly in the newswire domain and the biomedical domain. We review prior studies on both close-domain and open-domain event detection.

Closed Domain Event Detection

There is a substantial amount of prior work on event detection in the newswire and biology domain. Early studies on event detection used a rule-based algorithm. Grishman et al. (2005) present a baseline system called the Java Extraction Toolkit (JET)[16]. The system simply selects trigger instances from training data, and uses them in test data in order to activate their event extraction process. Most of prior work employs supervised token-level classifiers with various

[16] http://cs.nyu.edu/grishman/jet/jet.html

token-level and sentence-level features, such as word's surface form, lemma, part-of-speech tag, and its context words. Such models assign an event type or a non-event marker to each token. This formalization typically relies on the fact that the vast majority of triggers in an existing corpus consists of a single token. Ahn (2006) applies a logistic regression model to the binary classification of whether or not a word is an event trigger, and used a nearest-neighbor model for the multi-class classification to determine event types for the identified triggers. This two-stage approach achieved a F1 score of 60.1 on their own test data of the ACE 2005 corpus. Hardy et al. (2006) present a logistic regression model and two ensemble models, and obtained 59.76 classification accuracy for 11 event types in their corpus on the topic of weapons of mass destruction. More recent studies employed document-level features on top of the sentence-level features. Adapting the idea of "one sense per discourse" (Yarowsky, 1995) to event extraction, Ji and Grishman (2008) assume that how event triggers appear is consistent in topically-related documents. They collect clusters of the topically-related documents using a query based on triggers and arguments, and leverage document-level and cluster-level frequency statistics to implement the assumption. From an evaluation perspective, Ji and Grishman (2008) use a set of 40 newswire documents of the ACE 2005 corpus as a test set[17], and subsequent works follow this data split, using the same data set for system comparison. Table 3.17 shows a performance comparison between event trigger detection on the test data.

Model type	Learning type	System	P	R	F1
Feature-based	Supervised	MaxEnt (Ji and Grishman, 2008)	67.6	53.5	59.7
		CrossDoc (Ji and Grishman, 2008)	60.2	76.4	67.3
		CrossEvent (Liao and Grishman, 2010)	68.7	68.9	68.8
		CrossEntity (Hong et al., 2011)	72.9	64.3	68.3
		JointBeam (Li et al., 2013)	73.7	62.3	67.5
		Seed-based (Bronstein et al., 2015)	80.6	67.1	73.2
		JointEventEntity (Yang and Mitchell, 2016)	75.1	63.3	68.7
		PSL (Liu et al., 2016c)	75.3	64.4	69.4
		RBPB (Sha et al., 2016)	70.3	67.5	68.9
	Semi-supervised	PatternExpansion (Cao et al., 2015)	68.9	72.0	70.4
Neural network based	Supervised	CNN (Nguyen and Grishman, 2015)	70.2	65.2	67.6
		DMCNN (Chen et al., 2015)	75.6	63.6	69.1
		JRNN (Nguyen et al., 2016)	66.0	73.0	69.3
		FBRNN (Ghaeini et al., 2016)	66.8	68.0	67.4
		HNN (Feng et al., 2016)	84.6	64.9	73.4
	Semi-supervised	ANN-FN (Liu et al., 2016b)	77.6	65.2	70.7

Table 3.17: Comparison between reported performances of event trigger detection on the same ACE 2005 test set used in (Ji and Grishman, 2008). 'P' and 'R' stand for precision and recall, respectively.

Similarly to (Ji and Grishman, 2008), some prior works approach event detection by employing supervised classifiers such as logistic regression or support vector machines (SVMs) with

[17] See Section 2.1.1 for more details of the data split.

local features for token-level predictions and additional features based on document-level or cluster-level statistical information. Liao and Grishman (2010) present a self-training algorithm which expands training data by using an information retrieval system and global inference based on the cluster-level information of triggers and roles. Hong et al. (2011) make an assumption on the consistency of types of entity mentions co-occurred with triggers of a specific type and arguments of a specific role, and use the co-occurrence information as an additional feature for their SVM-based classifier. Cao et al. (2015) present a semi-supervised learning approach that expands training data by frequent patterns extracted from external corpora such as the English Gigaword corpus, and improve the performance of the JET system. Sha et al. (2016) propose another enhancement of the JET system, in which they employ trigger and sentence-level features as well as pattern features, and capture the relationships between candidate arguments using a maximum entropy classifier with regularization.

Recent works approach event detection as a sequence labeling problem, which is akin to our CRF model described in Section 3.4.1. Lu and Roth (2012) demonstrate the effectiveness of semi-Markov CRFs (Sarawagi and Cohen, 2004). Most of them took a pipelined approach where local classifiers identify triggers first, and then detect arguments. Li et al. (2013) present a incremental token-based structured perceptron model to detect triggers and arguments jointly using beam search. Similarly, joint dependencies in events were also addressed in the biomedical domain (Poon and Vanderwende, 2010; McClosky et al., 2011; Riedel and McCallum, 2011; Venugopal et al., 2014). Li et al. (2014) extend the model of (Li et al., 2013) by incorporating entity mentions and relations, and report that additional relation-event features and semantic frame features can improve the performance of event trigger detection. In a similar vein, Yang and Mitchell (2016) present a probabilistic graphical model that jointly extracts entity mentions, triggers, and arguments, using CRFs for extracting candidates of entity mentions and triggers in the first stage.

More recent works leverage neural networks for event detection. Nguyen and Grishman (2015) present a CNN that uses position and entity type features as additional embeddings. Chen et al. (2015) propose another variant of CNNs with a dynamic multi-pooling layer for event extraction. RNNs have been also explored for event detection. Nguyen et al. (2016) use a bidirectional GRU model to jointly predict event triggers and arguments. Ghaeini et al. (2016) leverage a forward-backward RNN (FBRNN) to process the left and right contexts of a trigger, and report the benefit of incorporating branch embeddings. Liu et al. (2016b) detect triggers in FrameNet by employing probabilistic soft logic, and use detected triggers to amplify training data for an Artificial Neural Networks (ANNs). Feng et al. (2016) develop a hybrid neural network combining RNN and CNN for event detection in English, Spanish and Chinese.

Ghaeini et al. (2016) also apply the FBRNN model described above to event nugget detection. Table 3.18 shows a comparison between state-of-the-art work, including FBRNN and top-ranked systems in the TAC KBP 2015 event nugget task. The top-ranked systems in the event nugget task have chosen different methods: CNN (Hong et al., 2015), a context-level neural embedding approach (Monahan et al., 2015; Reimers and Gurevych, 2015), CRFs (Liu et al., 2015), and an ensemble of MaxEnt, CRFs, a neural network model, and a seed-based methods (Luo et al., 2015).

System	Precision	Recall	F1
RPI_BLENDER (Hong et al., 2015)	75.23	47.74	58.41
LCC (Monahan et al., 2015)	73.95	46.61	57.18
LTI (Liu et al., 2015)	73.68	44.94	55.83
UKP (Reimers and Gurevych, 2015)	73.73	44.57	55.56
WIP (Luo et al., 2015)	71.06	43.50	53.97
FBRNN (Ghaeini et al., 2016)	71.58	48.19	57.61

Table 3.18: Comparison between reported performances of state-of-the-art systems for event nugget detection on TAC KBP 2015. The first five systems are the top five official submissions to the TAC KBP 2015 Event Nugget track.

Open Domain Event Detection

As compared to close-domain event detection described in Section 3.6.1, there are much less studies on open-domain event detection. Open IE (Banko et al., 2007; Fader et al., 2011) is an information extraction paradigm that is aimed at domain-independent discovery of relations extracted from text. Open IE is a relation-oriented formalism focusing on relation phrases, whereas our event definition is span-oriented. For instance, in Example (56) on page 56 ("His payment last wate"), 'was' a relation phrase whereas 'payment' is our event. Ritter et al. (2012) address open-domain event detection on Twitter. However, their event corpus is also manually annotated and thus small (1,000 tweets), and thus their model tends to be overfitting, as well as the supervised close-domain event detection models mentioned above. In contrast, our distantly-supervised method is not bound to any particular datasets, and a large number of synthesized examples allow the model to outperform supervised models, as shown in our experiments in Section 3.5.4.

3.6.2 Event Argument Detection

Similarly to event trigger detection, early studies leveraged statistical supervised classifiers to identify and classify event arguments. They typically formalize event argument detection as a subsequent step of event detection. They set up argument candidates by gold standard entity mentions in the same sentence as triggers obtained in the event detection step. Grishman et al. (2005) build two logistic regression classifiers: one for argument identification and the other for argument (role) classification. They did not report their system performance. Ahn (2006) implements a nearest-neighbor model and a logistic regression model, and the latter model obtained an F1 score of 57.3 on their own test data of the ACE 2005 corpus. Similarly to (Grishman et al., 2005), Ji and Grishman (2008) relied on the two logistic regression classifiers for argument identification and role classification, and employed the cluster-level statistics of argument distributions. Patwardhan and Riloff (2009) use a Naive Bayes classifier using local contextual features around a noun phrase. Liao and Grishman (2010) also present a system based on (Grishman et al., 2005), which improves argument classification by a self-training algorithm using global inference based on the cluster-level information of triggers and roles.

As described in Section 3.6.1, some researchers have explored joint models that simultaneously detect event arguments as well as event triggers. Using gold standard entity mentions as argument candidates, Li et al. (2013) propose an incremental token-based structured perceptron approach to predict event arguments in their output event graph structures. Li et al. (2014) extend this approach by incorporating entity mentions and relations, and report that additional relation-event features and semantic frame features can improve the performance of event argument detection. Yang and Mitchell (2016) present a probabilistic graphical model that extracts arguments jointly with triggers and entity mentions.

More recent works employ neural networks for event argument detection. Chen et al. (2015) propose a CNN-based model with a dynamic multi-pooling layer that utilizes context-word, position and event-type embeddings as additional features for argument classification. Nguyen et al. (2016) present a bidirectional GRU model that uses memory vectors and matrices to store the prediction information and predict arguments jointly with triggers.

3.6.3 Semi-supervised and Distantly-supervised Learning in NLP

Since data sparsity is a common problem in general, there is a considerable amount of prior work on semi-supervised and distantly-supervised learning. In this section, we focus on prior semi-supervised learning approaches used in NLP tasks.

Self-training (Scudder, 1965), also called bootstrapping, is a widely-used method for semi-supervised learning. First, it trains a classifier on existing labeled data. Second, it applies the classifier to unlabeled data, and generates additional labeled examples using the most confident predictions of the classifier. Third, it retrains the classifier on the expanded training data which comprises the original labeled data and the newly generated labeled data. These processes iterate until a certain condition is met (e.g., all unlabeled examples are labeled, or it reaches a preset number of iterations). This technique has been successfully applied to numerous NLP problems, including word sense disambiguation (e.g., (Yarowsky, 1995)), POS tagging (e.g., (Huang et al., 2009)), constituent parsing (e.g., (McClosky et al., 2006)), named entity recognition (e.g., (Daumé III, 2008)) and dependency parsing (e.g., (Wang et al., 2008)). Liao and Grishman (2011) present a self-training approach for event extraction by using an information retrieval technique. They argue that self-training with their logistic regression classifier does not perform very well for event detection for two reasons. First, the classifier uses its own predictions to train itself, and thus a classification mistake can reinforce itself. Second, nothing "novel" is added because the most confident examples are those frequently seen in the training data and might not provide "new" information. Huang and Riloff (2012) exploit role-identifying seed nouns for each event role, a collection of relevant (in-domain) and irrelevant (out-of-domain) texts and a semantic dictionary. Liu and Strzalkowski (2012) make use of dependency structure to create an event pattern for a particular event type, and present a method to derive new patterns by importing roles from another pattern.

Co-training (Blum and Mitchell, 1998) is another classic approach to semi-supervised learning, and can be viewed as an extension of self-training. First, it trains two classifiers on two different feature sets (views). Second, it applies the classifiers to unlabeled data, and generates additional labeled examples using the most confident predictions of each classifier. Third, it retrains the classifiers on the expanded training data, similarly to self-training. These processes

iterate until a certain condition is met, similarly to self-training. It has also shown improvement in a variety of NLP tasks, such as POS tagging (e.g., (Clark et al., 2003)), constituent parsing (e.g., (Sarkar, 2001)), and dependency parsing (e.g., (Sagae and Tsujii, 2007)).

Tri-training (Zhou and Li, 2005) is another extension of self-training. First, similarly to co-training, it trains three classifiers with different sets of features. Second, it applies the classifiers to unlabeled data, and generates additional labeled examples for a classifier if the other two classifiers agree on the labeling. Third, it retrains the classifiers on the expanded training data. These processes iterate until a certain condition is met, similarly to self-training and co-training. The tri-training algorithm iterates these processes. It is another successful technique which has been used in various NLP tasks, such as part-of-speech tagging (Søgaard, 2010), dependency parsing (Søgaard and Rishøj, 2010; Weiss et al., 2015) and CCG parsing (Lewis et al., 2016).

Distant supervision (Craven and Kumlien, 1999; Mintz et al., 2009) is a technique to generate additional labeled examples by matching the ground instances of a knowledge base to unlabeled text. In relation extraction, for example, Mintz et al. (2009) extract particular relations between two entities from Freebase, e.g., married(Barack Obama, Michelle Obama), and collect and label each pair of "Barack Obama" and "Michelle Obama" that appear in the same sentence as a positive example. Besides relation extraction, distant supervision has been successfully applied to a host of NLP tasks, including POS tagging (Hovy et al., 2015), named entity recognition (Ritter et al., 2011), passage retrieval (Xu et al., 2011), and semantic role labeling (Exner et al., 2015). In event detection, Reschke et al. (2014) apply distant supervision to extract airplane crash events from newswire corpora by casting an event as an n-ary relation. Liu et al. (2016b) use Probabilistic Soft Logic (Kimmig et al., 2012) to infer a mapping from frames in FrameNet to ACE event types, and improve their neural network model by using training data expanded with example sentences of frames in FrameNet via the mapping.

3.7 Summary

In this chapter, we described our approaches to closed-domain and open-domain event detection. In closed-domain event detection, we observed that even though closed-domain event detection focuses only on a particular subset of events in particular domains, supervised models cannot generalize well due to the overfitting problem, struggling with small training data. Therefore, we proposed a distant supervision approach to open-domain event detection in order to address both problems of restricted domains in event detection (Section 1.4.1) and data sparsity (Section 1.4.2). Due to the ubiquity and ambiguities of events, human annotation of events in the open domain is substantially expensive. Our distant supervision method is able to generate high-quality training data automatically, obviating the need for human annotation. The method is not bounded to any particular datasets and offers a versatile solution for event detection. Our experiment shows that the model outperforms supervised models in both in-domain and out-domain settings. This result indicates that the distant supervision enables robust event detection in various domains, while obviating the need for human annotation of events.

Chapter 4

Event Coreference Resolution

We defined two different types of event coreference, full coreference and partial coreference, based on a discussion about event identity in Section 1.3. This chapter discusses computational models for resolving full coreference (Section 4.1) and detecting subevents (Section 4.2). We describe related work on event coreference resolution in Section 4.3. Finally, we provide a summary of this chapter in Section 4.4. The work described in Section 4.2 is based on (Araki et al., 2014b).

4.1 Full Event Coreference Resolution

Full event coreference resolution is the task of determining whether two event mentions refer to the same event. In this thesis, we focus on closed-domain within-document event coreference resolution. As described in Section 1.5, one of our contributions is joint modeling for event detection and event coreference resolution, and we describe the contribution in detail in Chapter 5. In this section, we present our baseline event coreference models to be compared with the joint model while validating previous approaches in our setting.

In our formalization and model design for full event coreference resolution, we can deal with both event triggers and nuggets in the equivalent manner. Thus, we do not pay attention to distinguishing between event triggers and nuggets in this section. We regard full event coreference resolution as a problem of document-level structured prediction, and formalize it as follows. Given input document x with M gold standard event spans $\{m_j\}_{j=1}^{M}$ with an event type and a realis value, the goal of full event coreference resolution is to predict the correct event coreference clusters y for x. In this section, we first describe heuristic approaches in Section 4.1.1. We then present our two supervised methods for event coreference resolution: a latent antecedent tree model (Section 4.1.2) and a neural mention ranking model (Section 4.1.3).

4.1.1 Heuristic Approaches

It is known that *lemma match* is a simple yet strong heuristic approach for event coreference resolution (Yang et al., 2015; Choubey and Huang, 2017). This approach simply links event mentions which have the same lemmatized head word. In the case of closed-domain event detec-

tion for TAC KBP, it might be beneficial to employ other event attributes such as event types and realis in addition to lemma, given the TAC KBP definition of event coreference described in Section 1.3.1. An underlying assumption is that two coreferential event mentions are likely to have the same event type and realis in addition to the same lemmatized head word. To investigate the validness of the assumption, we first analyze the TAC KBP corpus. Table 4.1 shows distributions of event types and realis values over event coreference clusters in the dataset, respectively. For example, coreferential event mentions in 96.8% of event coreference clusters in the training data have the same event type. Similarly, coreferential event mentions in 88.1% of event coreference clusters in the training data have the same realis value. Hence, this analysis justifies the assumption described above adequately, and we develop our heuristics of lemma+type+realis match for the TAC KBP corpus.

		Train	Test
# documents		737	167
# event coreference clusters		2588 (100.0%)	605 (100.0%)
Event types	1 type	2505 (96.8%)	595 (98.3%)
	2 types	81 (3.1%)	10 (1.7%)
	3 types	2 (0.1%)	0 (0.0%)
Realis	A only or G only or O only	2280 (88.1%)	558 (92.2%)
	A only	1331 (51.4%)	322 (53.2%)
	G only	380 (14.7%)	81 (13.4%)
	O only	569 (22.0%)	155 (25.6%)
	A and G	37 (1.4%)	0 (0.0%)
	A and O	217 (8.4%)	47 (7.8%)
	G and G	43 (1.7%)	0 (0.0%)
	A, G and O	11 (0.4%)	0 (0.0%)

Table 4.1: Distributions of event types and realis values over event coreference clusters in the TAC KBP corpus. 'A', 'G' and 'O' stand for ACTUAL, GENERIC, and OTHER, respectively.

4.1.2 Latent Antecedent Tree Model

A *latent antecedent tree* (LAT) model is a feature-based approach which has been successfully applied to entity coreference resolution (Fernandes et al., 2012; Björkelund and Kuhn, 2014). This approach casts the problem of clustering coreferent mentions as constructing a latent tree representation. Each node of the tree represents a mention, and each edge denotes a coreference link between a mention and its most plausible antecedent. The tree first puts a dummy root node which does not represent any real mention. When processing a document in the linear order (from left to right), the LAT model creates a node for each mention and attaches it to the node of its antecedent mention if any, or attaches it to the root node if the mention has no antecedent (i.e., in the case of singletons). As a result, every subtree under the root node denotes a cluster of coreferent mentions. To train the model, we employ the structured perceptron algorithm (Collins, 2002), shown in Algorithm 2. In our formalization, a training example is

represented as a pair of input document x and its associated event graph y. Line 4 involves decoding to generate the best event graph for x. $\Phi(x, y)$ denotes a feature vector function that computes a feature vector for event graph y over x.

Algorithm 2 Structured perceptron.

Input: training examples $\{(x^{(k)}, y^{(k)})\}_{k=1}^{N}$
Input: number of iterations T
Output: weight vector $\mathbf{w}$
1: $\mathbf{w} \leftarrow \mathbf{0}$ ▷ Initialization.
2: **for** $t \leftarrow 1..T$ **do**
3: **for** $k \leftarrow 1..N$ **do**
4: $\hat{y}^{(k)} = \arg\max_{y \in \mathcal{Y}(x^{(k)})} \mathbf{w} \cdot \Phi(x^{(k)}, y)$
5: **if** $\hat{y}^{(k)} \neq y^{(k)}$ **then**
6: $\mathbf{w} \leftarrow \mathbf{w} + \Phi(x^{(k)}, y^{(k)}) - \Phi(x^{(k)}, \hat{y}^{(i)})$
7: **return** $\mathbf{w}$

Our system uses the following features: string match, part-of-speech combinations, and word embedding similarities. The motivation for the first two features are relatively obvious. As for the first feature, we assume that two event triggers with the same surface form are likely to corefer to the same event. With respect to the second feature, we assume that some particular pairs of part-of-speeches such as (verb, verb) and (noun, noun) will be a relatively strong indicator for event coreference. The string-match feature can be effective to some extent, but in general it is weak since event coreference can happen frequently with different lexical types such as paraphrases. To complement the weakness, we also devised the word embedding features to help the model resolve such event coreference. Word embeddings have been shown to capture lexico-semantic regularities; semantically similar words are close to each other in the embedding space (Agirre et al., 2009; Mikolov et al., 2013). Our assumption on the word embedding features is that if two lexically different event triggers corefer, their semantics should be still similar, and thus their corresponding word embeddings should be close to each other in the embedding space. For word embeddings, we use the pre-trained 300-dimensional word vectors from the Google News dataset (around 100 billion words) using word2vec tool[1], and apply cosine similarity as a numeric feature value to indicate how likely two event triggers corefer.

4.1.3 Neural Event Coreference Model

In addition to the LAT model, we also explore a neural mention ranking model for event coreference resolution. This model is largely based by the neural model for end-to-end entity coreference resolution by Lee et al. (2017) and forms the basis of our joint learning model that we will describe in Section 5.3. At the core of the model is a neural scoring module that computes an pairwise antecedent score, predicting how likely a preceding event nugget is an antecedent for each event nugget independently.

[1] https://code.google.com/p/word2vec/

Taking as input event nuggets detected in a prior step of event detection, the model performs event coreference resolution by employing two submodules. The first submodule constructs representation of event nuggets from their head word, as shown in Figure 4.1. More specifically, we first obtain input representation of the head word by concatenating its pre-trained embedding and its character representation produced by the character-level convolutional neural network (CNN) described in Section 3.4.2 (page 48). We then apply a BLSTM layer to the input representation and obtain head representation of event nuggets. Finally, we leverage event types and realis as additional features and concatenate the head representation with event type and realis embeddings, producing event representation:

$$\mathbf{e} = [\mathbf{h}; \mathbf{v}_t; \mathbf{v}_r] \tag{4.1}$$

where $\mathbf{h}$ is the head representation, and $\mathbf{v}_t$ and $\mathbf{v}_r$ are event type and realis embeddings, respectively. During training, the pre-trained word embeddings are fixed while both event type and realis embeddings are randomly initialized and fine-tuned.

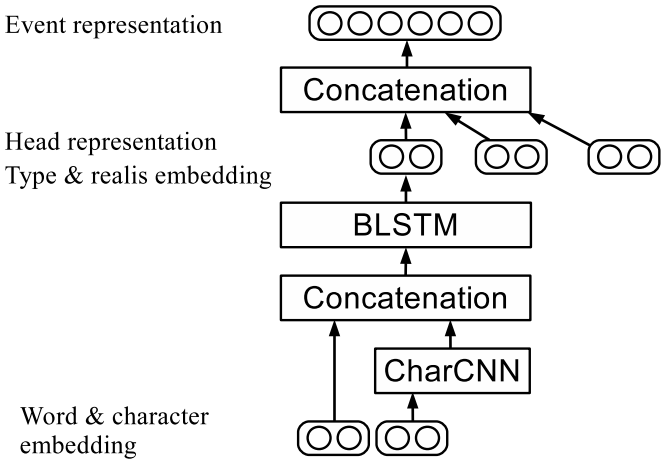


Figure 4.1: A high-level architecture of the first subnetwork to construct event representation.

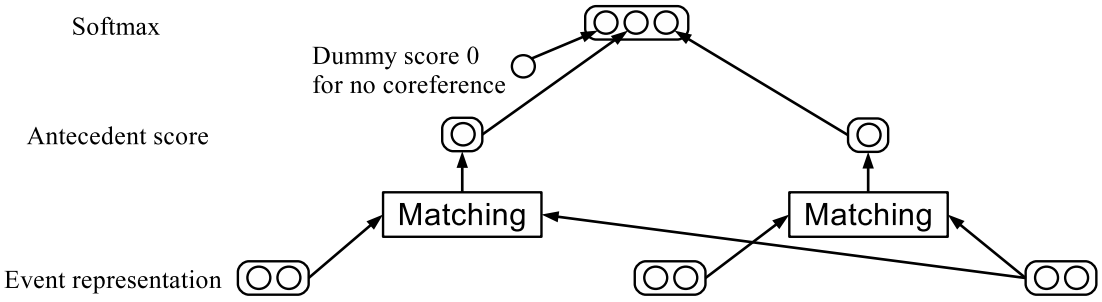


Figure 4.2: A high-level architecture of the second subnetwork to compute antecedent scores.

As shown in Figure 4.2, the second submodule takes the event representation as input and compute an antecedent score for each pair of an event nugget i and a preceding one j, using a heuristic matching technique inspired by (Mou et al., 2016):

$$\mathbf{m} = [\mathbf{e}_i; \mathbf{e}_j; |\mathbf{e}_i - \mathbf{e}_j|; \mathbf{e}_i \circ \mathbf{e}_j] \tag{4.2}$$

$$s_a(i, j) = \mathbf{w}_a \cdot \mathbf{m} + b_a \tag{4.3}$$

where $\cdot$ denotes the dot product and $\cdot$ denotes element-wise multiplication. We refer to the entire architecture as a neural event coreference (NEC) model.

The goal of this model is to assign an antecedent y_i to each event nugget i. The set of possible antecedent assignments for i is $\mathcal{Y}(i) = \{\epsilon, 1, \ldots, i-1\}$ where ϵ is a dummy antecedent denoting that i has no antecedent (no coreference), and $1, \ldots, i-1$ are all preceding event nuggets. As shown in Figure 4.2, we fix the antecedent score of ϵ to 0. With this formalization, the model decides event coreference if any non-dummy antecedent score is positive or decides no coreference (i.e., that i is a singleton) if all antecedent scores are negative.

Inference. A higher antecedent score $s_a(i, j)$ means that the preceding event nugget j is more likely to be an antecedent of i. The inference phase first seeks the best-scoring antecedent j^* for each event nugget i in a forward pass of a document. If the best-scoring antecedent has a positive score, the model forms event coreference between i and j^*. If it has a negative score, the model does nothing, leaving i as a singleton.

Learning. In the training phase, the loss function to minimize is the marginal negative log-likelihood of all correct antecedent assignments decided by gold standard event clusters. For each document, we compute:

$$L_{coref} = \sum_i \sum_{y \in \mathcal{Y}(i) \cap \mathrm{GOLD}(i)} \log \left(\mathrm{softmax}_y(s_a(i, j))\right) \tag{4.4}$$

$$= \sum_i \sum_{y \in \mathcal{Y}(i) \cap \mathrm{GOLD}(i)} \log \frac{\exp(s_a(i, y))}{\sum_{y' \in \mathcal{Y}(i)} \exp(s_a(i, y'))} \tag{4.5}$$

where $\mathrm{GOLD}(i)$ denotes the set of event nuggets in the gold event cluster containing i. If i is a singleton, $\mathrm{GOLD}(i) = \epsilon$.

Implementation Details. For word embeddings, we use the 300-dimensional GloVe vectors trained on a corpus of 42B words and do not fine-tune them during training. We map all out-of-vocabulary words to a zero vector. In CharCNN, we use character embeddings with 15 dimensions and 30 filters with window size 3. In BLSTM, we use one hidden layer with 500 units. We optimize model parameters using Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001 and a minibatch size of 1. We apply dropout (Srivastava et al., 2014) with 0.5 dropout rate to the word embeddings and the character representations from CharCNN. We train the model for up to 100 epochs, using early stopping based on performance on the validation set.

4.1.4 Experiments and Discussions

In our experiment of full event coreference resolution, we use the TAC KBP 2015 event nugget corpus introduced in Section 2.1.2. The input of a system in this task is spans and event types of

gold standard event nuggets. From the observation described in Section 4.1.1, we further added a constraint to the LAT model that coreferential event nuggets should share the same even type. Table 4.2 shows the official results of our system on the event hopper coreference task in the TAC KBP 2015 Event track.

Model	MUC	B^3	$CEAF_e$	BLANC	CoNLL Avg	TAC KBP Avg
LAT	68.33	79.72	71.49	49.38	66.86	67.23

Table 4.2: The official results (F1 scores) of our system on the event hopper coreference task.

Model	MUC	B^3	$CEAF_e$	BLANC	CoNLL Avg	TAC KBP Avg
Top 5	12.57	24.98	23.36	8.96	20.30	17.47
Top 4	19.30	28.66	28.64	13.56	25.53	22.54
Top 3	22.90	34.34	33.63	17.94	30.29	27.20
Top 2	33.79	39.88	35.73	26.06	36.47	33.87
Top 1	30.63	43.84	39.86	26.97	38.11	35.33
LTR (Baseline)	29.94	43.92	41.60	25.64	38.49	35.28
NEC-TR	30.19	44.38	42.88	26.17	39.15	35.91
NEC	**33.95**	**44.88**	**43.02**	**28.06**	**40.62**	**37.48**

Table 4.3: Performance (F1 scores) of event coreference resolution in the TAC KBP 2017 dataset. 'Top N' represents the Nth-ranked system reported in the official results.

Table 4.3 shows experimental results of event coreference resolution in the TAC KBP 2017 dataset. Note that unlike the previous experiment, no gold standard information is given in this experiment. Thus, we use our best-performing event detection system (i.e., BLSTM-MLC) described in Section 3.4.2 to obtain event nuggets before performing event coreference resolution. **LTR** is our baseline using the lemma+type+realis heuristics described in Section 4.1.1. **NEC** is our neural event coreference model described in Section 4.1.3, and **NEC-TR** is a variant of NEC where event representation does not utilize event type and realis embeddings, i.e., $\mathbf{e} = \mathbf{h}$ in Equation (4.1). As shown, the simple LTR baseline still achieves performance which is very close to that of the top system in TAC KBP 2017, thanks to the high performance of event detection.[2] As with Table 4.1 on page 72, this result indicates the importance role of event types and realis in event coreference resolution. On the other hand, NEC-TR does not utilize event types and realis at all, but still outperformed the performance of LTR and the top system. This result shows that the head representation itself is quite helpful along with the scoring architecture of predicting antecedent scores. The NEC model combines the advantages of LTR and NEC-TR adequately, outperforming the state-of-the-art performance of the top system in all the six metrics and by 2.2 TAC KBP F1 score.

4.2 Detecting Subevent Structures

In this section, we address the problem of *lack of subevent detection* stated in Section 1.4.3. Our approach is capable of detecting not only subevent parent-child relations but also subevent

[2] See Table 3.10 on page 55 for details of the event detection performance.

sister relations. To introduce the approach, we first define subevent structures (Section 4.2.1). We then present our two-stage approach for finding and improving subevent structures (Araki et al., 2014b). In the first stage, we introduce a multiclass event coreference resolver based on logistic regression, which is largely based on the pairwise coreference model proposed in the literature but can detect subevent parent-child and sister relations in addition to full coreference (Section 4.2.2). In the second stage, we pay attention to relatively good precision in the detection of subvent sister relations and propose a novel voting technique to improve the detection of subevent parent-child relations using subevent clusters detected in the first stage (Section 4.2.3).

4.2.1 Subevent Structures

We define a subevent structure as a hierarchical structure constructed by subevents and their parent. One aspect that makes event coreference resolution challenging is that events can relate to each other in various ways (Huttunen et al., 2002; Bejan and Harabagiu, 2008). In particular, some subevent relations exhibit subtle deviation from the full identity of events, as described in Section 1.3.2. Hence, detecting subevent structure is useful for event coreference resolution because we can reduce the difficulty of full coreference resolution by excluding subevent relations from candidates of full coreference chains after finding such structure.

We give Example (68) to illustrate a subevent structure. Figure 4.3 shows the subevent structure formed by the relations in Example (68).

(68) Ismail said the fighting, which lasted several days, intensified when forces loyal to Egal's Ha-bar Awal sub-clan of the Issak **attacked**(E77) a militia stronghold of his main opposition rival, ...

Egal militia, claiming to be the national defence force, said they had **captured**(E78) two opposition posts, **killing**(E79) and **wounding**(E80) many of the fighters, **destroying**(E81) three technicals (armed pick-up trucks) and **confiscating**(E82) artillery guns and assorted ammunition.

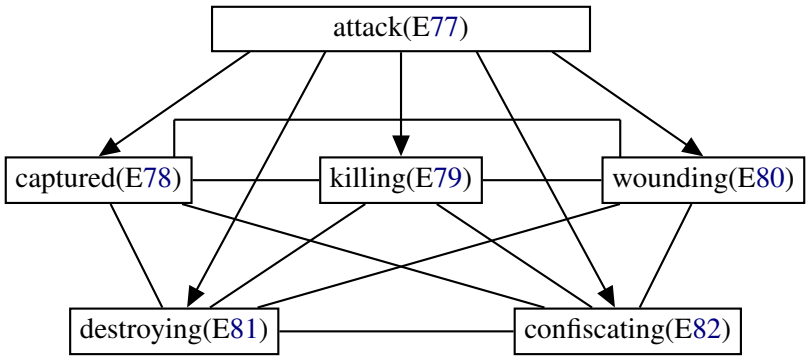


Figure 4.3: An example of subevent structure. An arrow represents a subevent parent-child relation with the direction from a parent to its subevent. A line represents a subevent sister relation between subevents under the same parent.

We see that E78, E79, E80, E81 and E82 form a cluster under their parent E77. Let us call

this cluster a *subevent cluster*. In our approach, we also pay attention to undirected relations between subevents sharing the same parent.

4.2.2 Event Relation Learning

Given that event mentions are annotated in a corpus, the goal of this stage is to build up a multiclass event coreference resolver that classifies a relation between two event mentions into one of the following four classes: full coreference (FC), subevent parent-child (SP), subevent sister (SS), or no coreference (NC). Our model is based on the pairwise coreference model (Chen et al., 2009b; Bengtson and Roth, 2008), which examines the relation between each pair of two event mentions. We use L2-regularized logistic regression to avoid overfitting. After training, it exclusively assigns one of the four classes above to each pair. We regard this model as our baseline system.

One additional note is that in the case of SP, our system internally models the directionality of that relation from the perspective of the discourse flow. Thus, it can output which event is a parent and which is its subevent, if necessary, in addition to an SP decision.

Features

Table 4.4 on page 79 lists our 135 features used in the logistic regression model. They can be organized into five groups as shown in the table. Our feature selection study showed that 'Subevent Ontology' and 'Narrative Schemas' are effective for SP and SS relations. As for the former feature, we developed a subevent ontology tree from our training data set, shown in Figure 4.5 on page 80. From the tree, we observed that some event words (e.g., 'raid' and 'explosion') show up as a subevent parent only, and others (e.g., 'kill' and 'injure') as a subevent only, while several words (e.g, 'attack' and 'bomb') can be both. Narrative schemas[3] aggregate structured sets of related events. Figure 4.4 on page 80 shows parts of the narrative schemas that are relevant to the Intelligence Community (IC) domain. We observed that the resource is particularly effective in capturing SS relations.

4.2.3 Subevent Detection

Our motivation for this stage comes from the result of the first stage. As we describe in Section 4.2.4, it turns out that our logistic regression model gains relatively high precision on SS relations. Therefore, we hypothesize that we can rely on the SS relations and resulting subevent clusters obtained in the first stage, and use a voting algorithm to select their parent for improving the system performance on SP relations.

The basic idea is that for each subevent cluster, we enumerate all event mentions (parent candidates) outside the cluster, and calculate probabilities of SP between each parent candidate and the cluster using the logistic regression model trained in the first stage. We then select out an event mention with the highest SP probability as the most likely parent for that cluster among the parent candidates. We consider two options for calculating the highest probability.

[3] http://www.usna.edu/Users/cs/nchamber/data/schemas/acl09/

Group	Feature Name	Description
Lexical (11)	Event String Similarity	String similarity measures between headwords of event mentions, including the Levenshtein distance, the Jaro coefficient, and the Dice coefficient.
	Modifier Similarity	The Dice coefficient between modifiers and event mentions.
Syntactic (44)	Part of Speech	Plurality, tense, nominality and verbality of headwords of event mentions.
	Syntactic Dependency	Dependency type between event mentions, annotated by the FANSE parser (Tratz and Hovy, 2011).
	Modifier Similarity	Whether event mentions are modified and whether headwords of event mentions are both modified by negation; the Dice coefficient of modifiers of event mentions (if both exist).
	Determiner	Whether a determiner of an event mention exists.
Semantic (41)	Subevent Ontology	Whether event mentions are in the subevent ontology constructed from the training data.
	Narrative Schemas	Scores given in the database of Narrative Schemas (Chambers and Jurafsky, 2009).
	Event as Entity	Whether nominal event mentions are resolved into entities by the Stanford coreference resolution system (Lee et al., 2011).
	WordNet Similarity	WordNet similarity scores between event mentions, including (Lesk, 1986), (Wu and Palmer, 1994), (Resnik, 1995), (Jiang and Conrath, 1997), (Hirst and St-Onge, 1998), (Leacock and Chodorow, 1998), and (Lin, 1998).
	SENNA Embeddings	The cosine similarity between word vectors for headwords of event mentions, given by the SENNA system (Collobert et al., 2011).
	Distributional Semantics	Whether event mentions are identical, decided by a semantic database of distributional semantic similarity between event mentions. The underlying model to compute distributional semantic similarity is described in (Goyal et al., 2013).
	VerbOcean	A score by VerbOcean (Chklovski and Pantel, 2004) as to a particular relation between head verbs of event mentions.
	Semantic Frame	Whether event mentions trigger the same semantic frame, extracted by SEMAFOR (Das et al., 2010).
	Mention Type	Whether event mentions have the same mention type, extracted the IBM SIRE system (Florian et al., 2010).
Semantic (arguments) (31)	Agent/Patient	Whether arguments are identical, decided by different matching algorithms (including the Stanford coreference resolution system and the Dice coefficient), and whether the numbers (e.g., 12 in 12 Somali) associated with arguments are identical.
	Location	Whether locations of event mentions are identical. This is decided by various matching algorithms, including the Dice coefficient, the Stanford coreference resolution system, and location subsumption (e.g., New York in the United States) using geographical knowledge bases such as DBpedia Mendes et al. (2012).
Discourse (8)	Sentence Distance	The number of sentences between two event mentions.
	Event Distance	The number of event mentions between two event mentions.
	Position	Whether an event mention is in the title or in the first sentence.

Table 4.4: A list of the features for our event relation learning. A number within parentheses in each feature group shows how many features belong to that group.

score=24.877186
Events: arrest kill shoot charge identify wound found endanger threaten harm
Scores: 6.440 6.149 5.657 5.174 4.690 4.617 4.407 4.283 4.255 4.034
...
score=16.74399
Events: destroy loot burn smash damage steal kill rip
Scores: 5.390 4.751 4.589 4.066 3.748 3.747 3.729 3.139
...
score=12.323438
Events: kill shoot wound ambush murder kidnap
Scores: 5.359 4.346 4.140 3.772 3.683 3.136
...

Figure 4.4: Excerpts from narrative schemas relevant to events in the IC domain. In each schema, the first line shows the overall score for that schema, and the third shows the individual verb scores, aligned with verbs in the second.

```
Root [350 (100.0)]
|-- attack [87 (24.9)]
|   |-- kill [30 (8.6)]
|   |-- wound [9 (2.6)]
|   |-- injure [6 (1.7)]
|   |-- fire [4 (1.1)]
...
|-- bomb [46 (13.1)]
|   |-- kill [17 (4.9)]
|   |-- injure [6 (1.7)]
|   |-- wound [4 (1.1)]
|   |-- explode [4 (1.1)]
...
|-- fight [24 (6.9)]
|   |-- kill [8 (2.3)]
|   |-- attack [5 (1.4)]
|   |-- wound [3 (0.9)]
|   |-- capture [1 (0.3)]
...
|-- raid [18 (5.1)]
|   |-- kill [4 (1.1)]
|   |-- arrest [3 (0.9)]
|   |-- bomb [1 (0.3)]
|   |-- setting [1 (0.3)]
...
```

Figure 4.5: Excerpts from the subevent ontology tree constructed from the training data set. The numbers in each node show a frequency of the headword of an event mention and its ratio (percentage) to the total number of occurrences of event mentions, which is 350. The tree shows subevent parents in the first level and subevent sisters in the second level.

In **Option 1**, we regard the highest probability as the highest SP probability among all pairs of parent candidates and sisters in the cluster. In **Option 2**, we sum up SP probabilities between a parent candidate and the sisters, and take the largest out of the sums.

4.2.4 Experiments and Discussions

In our experiment of subevent structure detection, we use the Intelligence Community (IC) corpus introduced in Section 2.1.3. We conduct 5-fold cross validation.

Evaluation

Unlike the experiment of full coreference resolution in Section 4.1.4, it is natural to use link-based metrics for evaluation since our system deals with four different relations between event mentions. Thus, we use BLANC (Recasens and Hovy, 2011), which claims that the metric is more adequate for coreference scoring. BLANC was developed to compute precision, recall, and the F1 score separately for two types of link (i.e., positive and negative links), and then average them for the final score. More specifically, if a system gains precision P_p and recall R_p for positive links, and precision P_n and recall R_n for negative ones, the BLANC F1 score is computed as follows:

$$F_{BLANC} = \frac{F_p + F_n}{2} = \frac{P_p R_p}{P_p + R_p} + \frac{P_n R_n}{P_n + R_n}$$

where F_p and F_n denote the F1 score for positive links and negative ones, respectively. Following the original definition, we apply BLANC to the four-class case as follows. Given system output as a 4x4 confusion matrix, we convert the matrix into four 2x2 one-vs-all confusion matrices, each of which represents a binary decision of the system as to each class. From these 2x2 matrices, we compute P_p, R_p, P_n, and R_n for each class, and then use them to compute F_{BLANC}.

Results

We constructed the logistic regression model using 135 features described in Section 4.2.2. We employ MetaCost (Domingos, 1999) to address the data imbalance shown in Table 2.7. Table 4.5 shows the system performance in the first stage. In this table, P and R stand for precision and recall, respectively, for positive and negative links, and F1 stands for the final BLANC F1 score. Table 4.5 indicates that the system achieved relatively high precision on SS relations in the first stage. This is basically because we incorporated more effective features for SS.

Table 4.6 shows the performance on SP relations in the second stage in terms of the BLANC scores. As compared to the baseline performance (the second row in Table 4.5), the second stage with Option 1 and 2 improved the BLANC F1 score by 2.5 points and by 3.2 points, respectively. We also see from Table 3 that Option 2 achieved a better performance than Option 1.

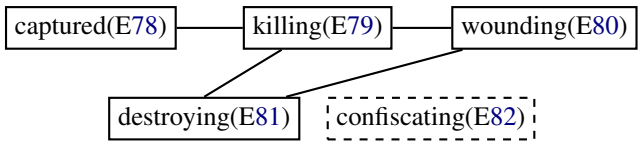
Figure 4.6(a) and Figure 4.6(b) illustrate how the system performs subevent structure detection through the two-stage process with respect to the subevent structure shown in Figure 4.3 on page 77. As shown in Figure 4.6(a), the extracted subevent cluster lost 'confiscating'(E82), but still captured four subevents out of the five in the gold standard. Figure 4.6(b) shows a subevent

Relation	Pos Links		Neg Links		Avg
	R	P	R	P	F1
FC	41.20	41.59	98.64	98.62	70.01
SP	**8.46**	**34.00**	**99.64**	**98.03**	**56.19**
SS	14.39	66.67	99.89	98.73	61.49
NC	98.18	95.36	23.92	45.24	64.02

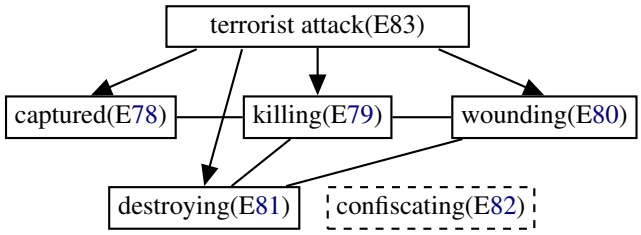
Table 4.5: BLANC scores gained in the first stage.

Relation	Pos Links		Neg Links		Avg
SP	R	P	R	P	F1
Option 1	13.43	31.03	99.35	98.13	58.74
Option 2	14.43	33.33	99.37	98.15	59.45

Table 4.6: BLANC scores gained in the second stage.



(a) The first stage.



(b) The second stage.

Figure 4.6: An example subevent structure detected in each stage. E82 is a missing event that the system fails to detect.

structure that the system obtained in the second stage from the subevent cluster extracted in the first stage. The system selected out 'terrorist attack'(E83) for a parent of the four subevents, which is different from 'attack'(E77) in Figure 4.3. However, E77 and E83 are fully coreferent in the gold standard annotation. Hence, all detected links in the subevent structure shown in Figure 4.6(b) are correct by means of extended subevent relations.

The comparison between Option 1 and 2 gives us an interesting insight on voting of subevents in an obtained cluster. Figure 4.7 provides an evidence to show where the performance difference between the two options comes from. In this figure, a numeric value stands for a subevent probability between a parent candidate and a subevent. Event trigger 'shootings'(E85) is the correct parent for the subevent cluster {E86, E87} in this case. The parent selection algorithm with Option 1 mistakenly chose 'violence'(E84) for the parent because the highest subevent probability 0.881 comes from the subevent relation between E84 and E86.

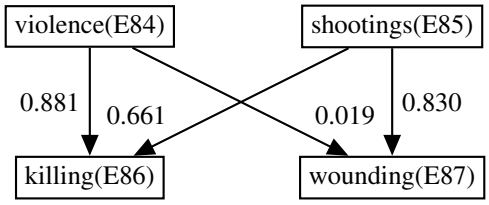


Figure 4.7: Parent selection from subevent sisters.

Our error analysis indicated that a common error derives from linguistic complexity in the expression of a subevent parent. For instance, in Example (69) below, E88 and E89 are subevents of E90. E90 is a rare, abstract term, making it difficult to capture SP relations.

(69) Over 90 Palestinians and one Israeli soldier have been **killed**(E88) since Israel **launched**(E89) a massive air and ground **offensive**(E90) into the Gaza Strip on June 28, ...

4.3 Related Work

Event coreference resolution is less studied as compared to the large body of work on event detection that we reviewed in Section 3.6.1. In this section, we review prior works on full event coreference resolution (Section 4.3.1) and subevent detection (Section 4.3.2).

4.3.1 Full Event Coreference Resolution

Full event coreference resolution is less studied than event detection. Early work performed event coreference resolution on scenario-specific events. Focusing on management succession (resignation) events, Humphreys et al. (1997) present a rule-based approach which utilizes a high-level ontology (semantic graph) whose nodes represent objects, events, or attributes. Bagga and Baldwin (1999) work on election and espionage events in addition to the resignation events, and present a cross-document event coreference system using a vector space model.

More recent work explores feature-based supervised learning models for event coreference resolution. Pradhan et al. (2007b) deal with both entity and event coreference in OntoNotes by

developing a generative model and two log-linear models. Chen et al. (2009b) propose a pairwise event coreference model based on maximum entropy with agglomerative clustering. Their experiments show that their argument and attribute features have a big impact on the performance of event coreference resolution. Bejan and Harabagiu (2010, 2014) present an unsupervised approach based on a class of nonparametric Bayesian models using a (potentially infinite) number of features to resolve both intra- and inter-document event coreference. Lee et al. (2012) form a system with deterministic layers to make coreference decisions iteratively while jointly resolving entity and event coreference. Sachan et al. (2015) describe an alternative approach to the joint entity and event coreference resolution using active learning, in which they cluster entity and event mentions by incorporating human judgment as pairwise constraints into their objective function. Liu et al. (2014) present a pairwise event coreference model, and show the benefit of information propagation between event mentions and their arguments in a post-processing step. Berant et al. (2014) propose a model that jointly predicts event arguments and event coreference (as well as other relations between event triggers).

There is very little work that uses neural networks for event coreference resolution. Krause et al. (2016) propose a convolutional neural network model that utilizes sentential features, inspired by recent studies on relation extraction (e.g., (dos Santos et al., 2015)) and event detection (e.g., (Nguyen and Grishman, 2015)). Aside from the machine learning based approaches described above, Lu and Ng (2016) leverage a rule-based approach called multi-pass sieves proposed by (Lee et al., 2013) for entity coreference resolution, and achieve an average F1 score of 40.32 on the TAC KBP event nugget corpus.

4.3.2 Subevent Detection

Detection of subevent parent-child relations is further less studied as compared to full event coreference resolution, and there is very little work on the task. Cybulska and Vossen (2012) present an unsupervised model to capture semantic relations and coreference resolution. Although their model considers non-full coreference in addition to full coreference, they do not show quantitatively how well their system performed in each of these two cases. Our work also differs from theirs in that we focused specifically on subevent parent-child relations while capturing subevent structure.

Script Induction

Script induction is the task of automatically inducing scripts from a large amount of text. This task shares the underlying motivation that stereotypical sequences of events are fundamental knowledge backbones to enable commonsense inferences in downstream applications. In fact, the task has attracted more research interest than subevent detection, yielding a larger body of work. The main focus of script induction is not on subevent parent-child relations but on event sequences. Methodologically speaking, prior work on script induction mostly simplifies events as N-tuple representations with arguments found by a dependency parser and models event sequences via event co-occurrences based on shared arguments. This formalization is not adequate for our subevent detection task, because the simplified event representation is very restrictive and subevent parents are often expressed without any arguments. Still, a reliable detection of

event sequences is important for subevent detection since the event sequences lend themselves to finding the parent-child relations, as we observed in Section 4.2.2.

One line of work approaches script induction in a participant-centric manner. Chambers and Jurafsky (2008) present an unsupervised learning approach for extracting partially ordered sets of events that all involve the same single participant, called narrative chains. In their work, an event slot is defined to be a tuple (v, d) where v is a verb and d is an argument slot (i.e., subject, object, or prep), and a narrative chain is represented as a tuple (L, O) where L is a set of event slots and O is a set of partial (temporal) orderings of verbs in L. An example of the narrative chain is: $L = \{(\text{hunt X}), (\text{X use}), (\text{suspect X}), (\text{accuse X}), (\text{search X})\}$ where X represents the argument slot, and $O = \{(\text{use}, \text{hunt}), (\text{suspect}, \text{search}), (\text{suspect}, \text{accuse})\}$. Chambers and Jurafsky (2009) extend the notion of narrative chains to represent coherent situation-specific sets of typed narrative chains, called narrative schemas. A typed narrative chain is defined as a triple (L, P, O) where L and O are defined as above, and P is a set of argument types (head words) representing a single role. An example of P corresponding to the above example of L and O is: $P = \{\text{person}, \text{company}, \text{criminal}\}$. They show that joint reasoning over event relations and argument roles enables a more informed prediction. Jans et al. (2012) find that the use of skip-grams and the ranking function based on n-gram probabilities by Manshadi et al. (2008) improves upon the performance of script prediction. Balasubramanian et al. (2013) show that the narrative schemas produced by Chambers and Jurafsky (2009) have several weaknesses, e.g., some schemas lack topical coherence and distinct roles are incorrectly merged into a single actor. They present a probabilistic model utilizing co-occurrence statistics of triples in the form of (Arg1, Relation, Arg2), and achieve more coherent schemas. In a similar vein, Pichotta and Mooney (2014) model the interactions between multiple arguments by incorporating coreference information into the model. Modi (2016) present a neural event sequence model based on (Modi and Titov, 2014), and they build distributed event representations using distributed representations of a predicate and its dependencies. Pichotta and Mooney (2016) employ Long Short-Term Memory (LSTM) for modeling scripts, and present two different ways to train the model, according to whether to use noun information or coreference information for event arguments.

Another line of work approaches script induction from a temporal perspective, which aligns with our work better. Regneri et al. (2010) focus on what phrases can describe the same event in a script, and what constraints must hold on the temporal order of events. They propose a multiple sequence alignment algorithm to build a graph representation of temporal event structure of scripts, and evaluate the algorithm using a corpus of 493 human-annotated event sequence descriptions (ESDs) for 22 scenarios. Frermann et al. (2014) leverage a hierarchical Bayesian model to jointly induce event sequences and constraints on their orderings from sets of ESDs. Modi and Titov (2014) propose a neural compositional model to construct event representations based on distributed representations of predicates and their arguments, and then use the representations to predict event orderings in the ESDs.

4.4 Summary

We have presented computational models for full event coreference resolution and subevent structure detection. In the former task, we presented a document-level latent tree model and

a neural mention ranking model for event coreference resolution. Running on top of the output of our best-performing event detection model, the neural model achieved the state-of-the-art performance in the event coreference resolution task of TAC KBP 2017.

In the latter task, we proposed a multiclass logistic regression model that can detect subevent relations in addition to full coreference. We then proposed and evaluated a novel approach to improve subevent structure using a voting algorithm. Our evaluation indicates that the approach achieves significantly better performance gain. To the best of our knowledge, this is the first work to differentiate subevent relations as partial coreference from full coreference, and examine subevent structure including subevent sister relations. One possible extension to this work is to systematically check structural consistency beyond pairwise decisions and resolve inconsistency in detected subevent structures, thereby obtaining a better performance on SP and SS. In addition, we can construct a library of domain event backbones by aggregating improved subevent structures, and then use it as a background knowledge resource for resolving full coreference in related domains.

Chapter 5

Joint Modeling for Event Detection and Event Coreference Resolution

In Chapter 3 and Chapter 4, we addressed the tasks of event detection and event coreference resolution separately. In this chapter, we present our approaches for jointly extracting events and resolving event coreferences in order to address the issue on error propagation in pipeline models described in Section 1.4.4. We first describe the issue in more detail in Section 5.1, and propose two different joint models: a feature-based model (Section 5.2) and a neural network based model (Section 5.3). We also review prior studies on joint structured learning in NLP in Section 5.4. Lastly, we put a summary of this chapter in Section 5.5. The work described in Section 5.2 is based on (Araki and Mitamura, 2015).

5.1 Event Interdependencies via Event Coreference

As described in Section 1.2.3, we have defined an event to be an eventuality which involves zero or more of participants, attributes, or both. Their salient property among other linguistic phenomena is that they compose rich semantic and discourse structures. On the one hand, they involve various participants and attributes locally, often within a sentence, forming a minimal unit of factual information: who did what to whom where and when. On the other hand, they corefer to each other globally, often across sentences, playing a role of discourse connection points to form a coherent story. Both views are indispensable to natural language understanding.

These semantic and discourse aspects of events are not independent from each other, and in fact often work in interactive manners. Let us refer to such event interdependencies via event coreference as *event interaction*. We give six examples of the event interaction:

(70) Trebian was **born**$_{\text{Be-Born}}$(E91) on November 4th. We were praying that his father would get here on time, but unfortunately he missed **it**$_{\text{Be-Born}}$(E92).

(71) It's important that people all over the world know that we don't believe in the **war**$_{\text{Attack}}$(E93). Nobody questions whether **this**$_{\text{Attack}}$(E94) is right or not.

(72) Mary is not the only military wife who received a photo and **letter**(E95). Her friend Anna also got **one**(E96).
Phone-Write Phone-Write

(73) In a village near the West Bank town of Qalqiliya, an 11-year-old Palestinian boy was **killed**(E97) during an exchange of **gunfire**(E98). Also Monday, Israeli soldiers **fired**(E99) on four diplomatic vehicles in the northern Gaza town of Beit Hanoun, diplomats said. There were no **injuries**(E100) from the **incident**(E101).
Die Attack Attack Injure Attack

(74) The slogan makes for a powerful bumper sticker – guns don't **kill**(E102) people, people **do**(E103)! But this next story makes the case the slogan doesn't go far enough. A better one might be, people **do**(E104) and people **die**(E105) and still other people search their souls and struggle for redemption.
Die Die Die Die

(75) A car bomb that police said was set by Shining Path guerrillas **ripped off**(E106) the front of a Lima police station before dawn Thursday, **wounding**(E107) 25 people. The **attack**(E108) marked the return to the spotlight of the feared Maoist group, recently overshadowed by a smaller rival band of rebels. The pre-dawn **bombing**(E109) **destroyed**(E110) part of the police station and a municipal office in Lima's industrial suburb of Ate-Vitarte, **wounding**(E111) 8 police officers, one seriously, Interior Minister Cesar Saucedo told reporters. The bomb **collapsed**(E112) the roof of a neighboring hospital, **injuring**(E113) 15, and **blew out**(E114) windows and doors in a public market, **wounding**(E115) two guards.
Attack Injure Attack Attack Attack Injure Attack Injure Attack Injure

Each of the first five examples has an event coreference: E91-E92, E93-E94, E95-E96, and E98-E101, E103-E104. E92, E94 and E96 are a pronoun, and E101 is a common noun with a somewhat abstract meaning. Thus, they cannot exhibit strong event semantics associated with a particular event type by themselves. The pronouns may seem to refer to an entity rather than an event. In fact, state-of-the-art coreference (pronoun) resolvers cannot be helpful to these cases mainly because they are trained for resolving entities. For instance, we ran the Stanford coreference resolver[1] on these examples, but it resolves none of the event coreferences correctly. E104 is an auxiliary verb, and also has less evidence of associating the trigger with a specific type. Therefore, E92, E94, E96, E101 and E104 are relatively difficult to be recognized as triggers of a particular event type by themselves. However, the event coreferences E91-E92, E93-E94, E95-E96, E98-E101 and E103-E104 help to detect E92, E94, E96, E101 and E104, respectively. On the other hand, previous works typically rely on a pipelined model that extracts triggers at the first stage, and then resolves event coreference at the second stage. Although this modularity is preferable from development perspectives, the pipelined model limits the event interaction. Namely, the first stage alone is unlikely to detect E92, E94, E96, E101 and E104 due

[1] http://nlp.stanford.edu/software/dcoref.shtml

to the difficulties described above. These missing events make it impossible for the second stage to resolve the event coreferences E91-E92, E93-E94, E95-E96, E98-E101 and E103-E104.

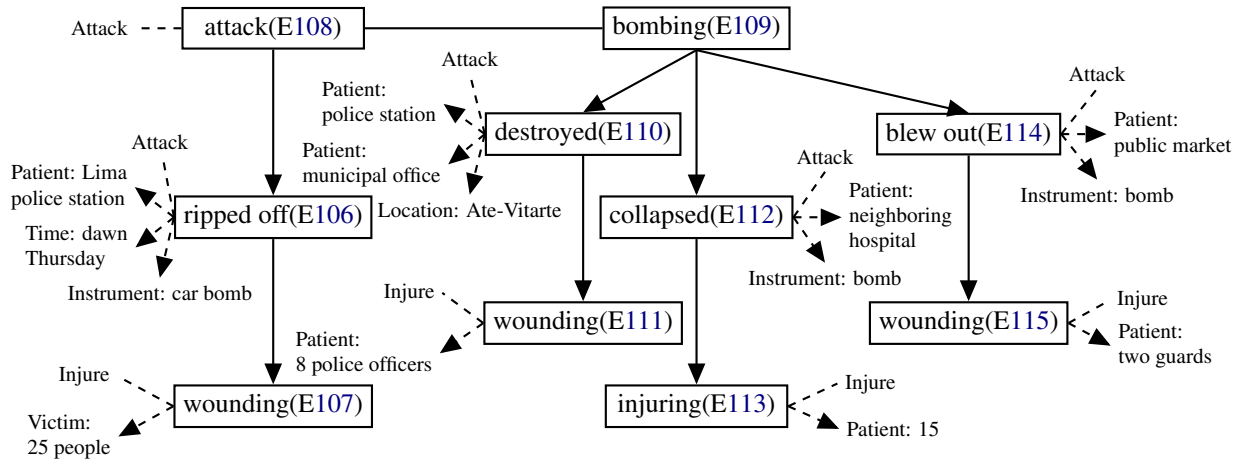


Figure 5.1: An event structure for Example (75) in the newswire domain. Dashed straight lines and arrows represent event types and event arguments, respectively. Solid straight lines and arrows represent event coreferences and subevents, respectively.

Example (75) on page 88 is a more complex example to illustrate how the event interaction matters. For clarification, we show an event structure of this example in Figure 5.1. This is a fairly complex event structure, but if one first focuses only on event words and phrases, one could extract E110 and E111 because they have somewhat discernible syntax forms (i.e., verbs) and lexical semantics. Once these are extracted, one can incorporate the knowledge of a typical discourse flow {bombing $\rightarrow$ destruction $\rightarrow$ wounding} in the terrorism domain. This would be of much help to differentiate between a big bombing event E108, its descendants E106 and E107, E109 (coreferent with E108), and its three subevents E110, E112 and E114, and their descendants E111, E113 and E115, as well as the numbers of wounded people of E111, E113 and E115: 8 police officers, 15 (people), and two guards. This global disambiguation then facilitates further local event extraction. As shown in this example, logical combinations of event interactions compose event structure. These observations motivate our approach to jointly extracting events and resolving event coreferences.

5.2 Joint Modeling with Feature-based Models

To address the problem of the event interdependencies described in Section 5.1, we present a document-level, feature-based structured learning model that simultaneously detects events and resolves event coreferences (Araki and Mitamura, 2015). We formalize the extraction of event triggers and event coreference as a problem of structured prediction. The output structure is a document-level event graph where each node represents an event trigger, and each edge represents an event coreference link between two event triggers.

1. Whether to end with a suffix '-tion' or '-ment'
2. Whether to begin with a determiner
3. The number of tokens in the trigger
4. Lemma of the head word
5. Lower-case string of the trigger
6. Part-of-speech tag of the head word
7. All part-of-speech tags of the trigger
8. Relation type of dependency heads
9. Relation type of dependency children
10. Semantic role types if the trigger is a predicate
11. Levin verb classes of the head word
12. Whether the trigger is a title of pages under category 'Biological processes' and related categories

Table 5.1: A list of features for event trigger identification.

5.2.1 Event Graph Learning

As with the formalization described in Section 3.4, let x denote an input document with n tokens where x_i is the i-th token in the document. For event graph learning, we use structured perceptron (Collins, 2002) shown in Algorithm 2 on page 73, and average weights to reduce overfitting as suggested in (Collins, 2002). The algorithm involves decoding to generate the best event graph for each input document. We elaborate on our decoding algorithm in Section 5.2.2. Since an event graph has an exponentially large search space, we use beam search to approximate exact inference. We extract a range of features by using Stanford CoreNLP (Manning et al., 2014), MATE (Björkelund et al., 2009), OpenNLP[2], Nomlex (Macleod et al., 1998), and Levin verb classes (Levin, 1993). We provide details of our features for event trigger identification in Table 5.1 and those for event coreference resolution in Table 5.2.

We use the standard-update strategy in our structured perceptron model. As variants of structured perceptron, one could employ the early update (Collins and Roark, 2004) and max-violation update (Huang et al., 2012) to our model. Our initial experiments indicated that early updates happen too early to gain sufficient feedback on weights from entire documents in training examples, ending up with a poorer performance than the standard update. This contrasts with the fact that the early-update strategy was successfully applied to other NLP tasks such as constituent parsing (Collins and Roark, 2004) and dependency parsing (Zhang and Clark, 2008b). The main reason why the early update fell short of the standard update in our setting is that joint event trigger identification and event coreference resolution is a much more difficult task since they require more complex knowledge and argument structures. Due to the difficultly of the task, it is also very difficult to develop such an effective feature set that beam search can explore the search space of an entire document thoroughly with early updates. This observation follows (Björkelund and Kuhn, 2014) on entity coreference resolution. In contrast, the max-violation update showed almost the same performance as the standard update on the development data. From these results, we chose the standard-update strategy for simplicity.

[2] http://opennlp.apache.org/

1. Whether 3-character prefixes match
2. Whether Lower-case strings match
3. Whether Head word strings match
4. Whether Lemma of head words match
5. Whether the Nomlex nominalization of head words match
6. Whether the following trigger is an acronym of the preceding trigger
7. Whether the following trigger is in apposition to the preceding trigger
8. Part-of-speeches of head words
9. Dependency path between the triggers
10. Whether the following trigger has a determiner
11. Whether both triggers are predicates
12. Whether both triggers share a semantic role
13. Whether both triggers has an argument of the same string with the same semantic role
14. Levin verb classes which both head words belong to
15. Whether the preceding trigger is the first word in a document

Table 5.2: A list of features for event coreference resolution.

5.2.2 Joint Decoding

Given that an event trigger has one or more tokens, event trigger identification could be solved as a token-level sequential labeling problem with the BIO or BILOU encoding scheme in the same way as named entity recognition (Ratinov and Roth, 2009). If one uses this approach, a beam state may represent a partial assignment of an event trigger. However, event coreference can be explored only from complete assignments of an event trigger. Thus, one would need to synchronize the search process of event coreference by comparing event coreferences from the complete assignment at a certain position with those from complete assignments at following positions. This makes it complicated to implement the formalization of token-level sequential labeling for joint decoding in our task. One possible way to avoid this problem is to extract event trigger candidates with a preference on high recall first, and then search event coreference from those candidates, regarding them as complete assignments of an event trigger. This recall-oriented pre-filtering is often used in entity coreference resolution (Lee et al., 2013; Björkelund and Farkas, 2012). In our initial experiments, we observed that our rule-based filter gained around 97% recall, but extracted around 12,400 false positives against 823 true positives in the training data. This made it difficult for our structured perceptron to learn event triggers, which underperformed on event coreference resolution.

We, therefore, employ segment-based decoding with multiple-beam search (Zhang and Clark, 2008a; Li and Ji, 2014) for event trigger identification, and combine it with the best-first clustering (Ng and Cardie, 2002) for event coreference resolution in document-level joint decoding. The key idea of segment-based decoding with multiple-beam search is to keep previous beam states available, and use them to form segments from previous positions to the current position. Let l_{max} denote the upper bound on the number of tokens in one event trigger. The k-best partial

Algorithm 3 Joint decoding for event triggers and coreference with beam search.

Input: input document $x = (x_1, x_2, \ldots, x_n)$
Input: beam width k, max length of event trigger l_{max}
Output: best event graph $\hat{y}$ for x

```
1: initialize empty beam history B[1..n]
2: for i ← 1..n do
3:   for l ← 1..l_max do
4:     for y ∈ B[i − l] do
5:       e ← CreateEventTrigger(l, i).
6:       AppendEventTrigger(y, e)
7:       B[i] ← k-Best(B[i] ∪ y)
8:       for j ← 1..i − 1 do
9:         c ← CreateEventCoref(j, e).
10:        AddEventCoref(y, c)
11:        B[i] ← k-Best(B[i] ∪ y)
12: return B[n][0]
```

structures (event subgraphs) in beam B at the j-th token is computed as follows:

$$B[j] = \underset{y \in \{y_{[1:j-l]} \in B[j-l],\ y_{[j-l+1,j]} = s\}}{k\text{-BEST}} \Phi(x, y) \cdot \mathbf{w}$$

where $1 \leq l \leq l_{max}$, $y_{[1:j]}$ is an event subgraph ending at the j-th token, and $y_{[j-l+1,j]} = s$ means that partial structure $y_{[j-l+1,j]}$ is a segment, i.e., an event trigger candidate with a subsequence of tokens $x_{[j-l+1,j]}$. This approximates Viterbi decoding with beam search.

The best-first clustering incrementally makes coreference decisions by selecting the most likely antecedent for each trigger. Our joint decoding algorithm makes use of the incremental process to combine the segment-based decoding and best-first clustering. Algorithm 3 shows the summary of the joint decoding algorithm. We give explanations of four functions used in the algorithm as follows:

- CreateEventTrigger(l, i) is a function that newly creates an event trigger of length l at index i.
- AppendEventTrigger(y, e) is a function that adds event trigger e to event graph y as a new node.
- CreateEventCoref(j, e) is a function that newly creates an event coreference link between an event nugget at index j and event nugget e.
- AddEventCoref(y, c) is a function that adds event coreference link c to event graph y as a new edge.

Line 3-7 implements the segment-based decoding, and line 8-11 implements the best-first clustering. Once a new event trigger is appended to an event subgraph at line 6, the decoder uses it as a referring mention regardless of whether the event subgraph is in the beam, and seeks the best antecedent for it. This enables the joint model to make a more global decision on event trigger identification and event coreference decision.

5.2.3 Experiments and Discussions

We describe our experiments of joint learning of feature-based models in this section.

Corpus

In our experiments, we use the ProcessBank introduced in Section 2.1.4. The original corpus provides 150 paragraphs as training data, and we split them into 120 and 30 for our training and development, respectively. We chose ProcessBank instead of a larger corpus such as the ACE 2005 corpus for the following two reasons. First, the human annotation of event coreference links in ProcessBank enables us to apply the best-first clustering directly; on the other hand, this is not readily feasible in ACE 2005 since it annotates event coreference as clusters, and gold standard event coreference links required for the best-first clustering are not available. Second, event coreference resolution using ProcessBank is novel since almost no previous work on the task used that corpus. The only exception could be (Berant et al., 2014), where they extracted several types of relations between event triggers, including event coreference. However, they did not report any performance scores of their system specifically on event coreference, and thus their work is not comparable to ours.

ProcessBank annotates multi-token events in addition to single-token ones, but they are all continuous. ProcessBank does not assign any domain-specific event types to events. In that sense, ProcessBank's scheme for annotating events is close to ACE's event triggers without any types. Unlike previous work on joint modeling (Berant et al., 2014; Li et al., 2013), we explicitly allow an event trigger to have multiple tokens, such as verb phrase 'look into' and compound proper noun 'World War II'. This is a more realistic setting for event trigger identification, since in general there are a considerable number of multi-token event triggers.[3]

When training our model, we observed that 20-iteration training almost reached convergence, and thus we set the number of iterations to 20. We set l_{max} to 6 because we observed that the longest event trigger in the entire ProcessBank corpus has six tokens. When tuning beam width k on the development set, large beam width did not give us a significant performance difference. We attribute this result to the small size of the development data. In particular, the development data has only 28 event coreferences, which makes it difficult to reveal the effect of beam width. We thus set k to 1 in our experiments.

Baseline Systems

Our baseline is a pipelined model that divides the event trigger decoding and event coreference decoding in Algorithm 3 on page 92 into two separate stages. It uses the same structured perceptron with the same hyperparameters and feature templates. We choose this baseline because it clearly reveals the effectiveness of the joint model by focusing only on the architectural difference. We also compare our system with **ProRead** (Berant et al., 2014), which is the state-of-the-art system for event coreference resolution in the ProcessBank corpus.

[3]For example, around 13.4% of the 1403 event triggers in ProcessBank have multiple tokens.

Results

We first show the result of event coreference resolution in the test data in Table 5.3. The joint model outperforms the baseline by 6.9 BLANC F1 and 1.8 CoNLL F1 points. We observed that this overall performance gain comes largely from a precision gain, more specifically, substantially reduced false positives. We explain the superiority of the joint model as follows. In the baseline, the second stage uses the output of the first stage. Since event triggers are fixed at this point, the baseline explores coreference links only between these event triggers. In contrast, the joint model seeks event triggers and event coreference simultaneously, and thus it explores a larger number of false positives in the search process, thereby learning to penalize false positives more adequately than the baseline.

Model	MUC	B^3	$CEAF_e$	BLANC	CoNLL Avg	TAC KBP Avg
Pipeline (Baseline)	22.53	57.01	57.44	25.05	45.66	40.51
ProRead	12.50	63.81	64.18	27.54	46.83	42.01
Joint (Ours)	26.08	57.93	58.38	31.91	**47.45**	**43.58**

Table 5.3: Results (F1 scores) of event coreference resolution.

System	Precision	Recall	F1
Pipeline (Baseline)	64.85	57.02	60.68
ProRead	67.82	65.73	**66.76**
Joint (Ours)	65.24	55.89	60.21

Table 5.4: Results of event trigger identification. 'Baseline' refers to the first stage of our baseline.

Table 5.4 shows the results of event trigger identification on the test data. Similarly in event coreference resolution, we observed that the joint model also achieved a precision gain deriving from a reduction of false positives. However, its improvement on precision is small, ending up with almost the same F1 point as the baseline. We speculate that this is due to the small size of the corpus, and the joint model was unable to show its advantages in event trigger identification.

Example (76) and Example (77) below are two error cases in event coreference resolution. Our model fails to resolve E116-E117 in Example (76) and E118-E119 in Example (77). The model was unable to adequately extract features for both event triggers and event coreference, particularly because their surface strings are not present in training data, they are lexically and syntactically different, and they do not share key semantic roles (e.g., agents and patients) in a clear argument structure.

(76) When the cell is stimulated, gated channels open that facilitate Na+ **diffusion**(E116). Sodium ions then **"fall"**(E117) down their electrochemical gradient, ...

(77) The next seven steps **decompose**(E118) the citrate back to oxaloacetate. It is this **regeneration**(E119) of oxaloacetate that makes this process a cycle.

5.3 Joint Modeling with Neural Models

In this section, we describe our joint modeling based on neural network models. The approach is aimed to improve the performance of event detection and event coreference resolution in the TAC KBP corpus. The key assumption of the joint modeling is that we can improve recall in both event detection and event coreference resolution by exploring more possibilities of event detection and event coreference resolution while not committing to single output from event detection models.

5.3.1 Joint Decoding

We first present a relatively simple approach for joint modeling: joint decoding. We refer to this method as **JD**. The main idea of JD is that we allow for low-scoring event nuggets to some extent and discover coreferential event nuggets among a larger set of event nuggets than the pipeline approach, thereby improving recall. A sequence pipeline of our neural event detection model (BLSTM-MLC in Section 3.4.2), realis model (BLSTM+CharCNN in Section 3.4.3) and event coreference model (NEC in Section 4.1.3) outperformed the state-of-the-art performance in the TAC KBP dataset, as described in Section 3.4.4 and Section 4.1.4. We use these individually pre-trained models in the joint decoding method.

As described in Section 3.4.4 (page 52), BLSTM-MLC optimizes event detection performance by finding the best probability threshold to cut off low-scoring type predictions. Let p_t denote the probability threshold. In the joint decoding method, we use two probability thresholds instead:

- p_c: a threshold for cut off coreferential event nuggets.
- p_s: a threshold for cut off singleton event nuggets.

As for p_c, we choose a lenient probability threshold where $p_c \leq p_t$ and leave those low-scoring type predictions for further consideration of event coreference. For example, if BLSTM-MLC finds $p_t = 0.41$ for the best probability threshold, we heuristically consider range $\{0.20, 0.21, \ldots, 0.41\}$ for p_c. We then let NEC perform event coreference resolution against a larger number of event nuggets including low-scoring ones whose probability is in $[p_c, p_s]$. If event coreference is found, we keep all event nuggets involving the event coreference. If not, such event nuggets end up with singletons, and we prune them if their probability is lower than p_s. We tune p_c and p_s on the validation set. Our assumption of joint decoding is that some low-scoring event nuggets, particularly the ones whose probability is lower than but close to p_t, may be correct event nuggets, and among them those referring to antecedents can be recovered by event coreference decisions, thereby improving recall.

As for realis prediction, we simply use the BLSTM+CharCNN model to predict realis values of all detected event nuggets before NEC performs event coreference resolution.

5.3.2 Joint Training

We propose another method for joint modeling, which is is joint training. We refer to this method as **JT**. In this method, we jointly train BLSTM-MLC and NEC models by sharing low-level layers of the networks. Figure 5.2 shows a high-level overview of the joint neural architecture.

The architecture for joint training is inspired by the idea of multi-task learning; our assumption on joint training is that training signals from related tasks bring about superior regularization for neural network models.

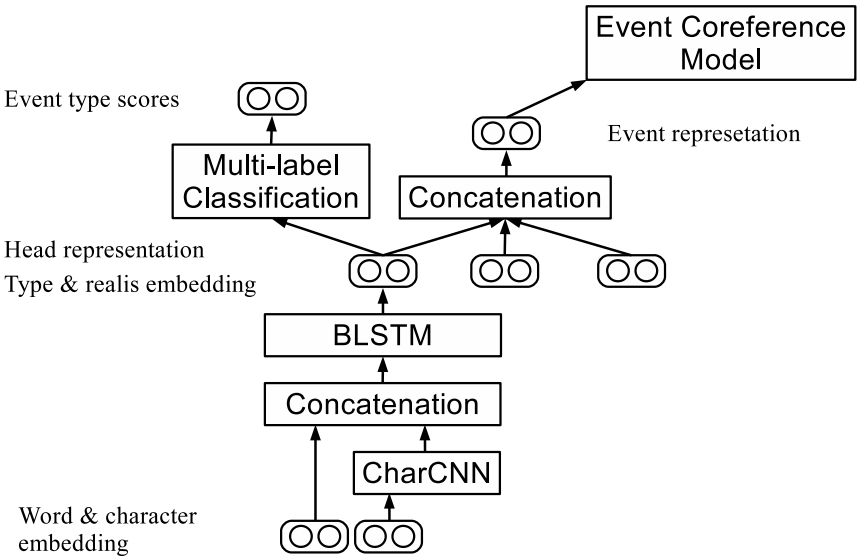


Figure 5.2: A neural architecture for joint training of event detection and event coreference resolution.

During training, we minimize the sum of the loss of BLSTM-MLC (L_{mlc} in Equation (3.14) on page 50) and that of NEC (L_{coref} in Equation (4.4) on page 75):

$$L_{joint} = L_{mlc} + L_{coref} \tag{5.1}$$

In the inference phase, we employ the joint decoding technique described in Section 5.3.1 and use the pre-trained realis model (BLSTM+CharCNN in Section 3.4.3) for realis prediction.

Implementation Details. We use the 300-dimensional GloVe vectors trained on a corpus of 42B words[4] from Pennington et al. (2014) and do not fine-tune them during training. We map all out-of-vocabulary words to a zero vector. In CharCNN, we use character embeddings with 15 dimensions and 30 filters with window size 3. In BLSTM, we use one hidden layer with 1000 units. For type prediction, we put a feedforward neural network which has one hidden layer with 500 dimensions and rectified linear units (Nair and Hinton, 2010) for non-linear activation. We set the dimensiionality of both event type embeddings and realis embeddings to 50. We optimize model parameters using Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001. We use a minibatch size of 32 for event type and realis prediction and a minibatch size of 1 for event coreference resolution. We apply dropout (Srivastava et al., 2014) with 0.5 dropout rate to the word embeddings and the character representations from CharCNN. We also apply dropout with 0.2 dropout rate to the hidden layer of BLSTM. We train the model for up to 100 epochs, using early stopping based on performance on the validation set.

[4] https://nlp.stanford.edu/projects/glove/.

5.3.3 Experiments and Discussions

Table 5.5 and Table 5.6 show the results of event type prediction (span+type) and overall event detection (span+type+realis), respectively. For comparison, Table 5.5 also reports our previous result shown in Table 3.6 on page 53, and Table 5.6 reports the one shown in Table 3.10 on page 53. As shown, our joint modeling methods improve recall of event detection and makes a further improvement on the performance. The performance difference between JT+JD and BLSTM-MLC is statistical significant at $p < 0.05$, based on a two-tailed paired t-test.

Model	Precision	Recall	F1
Top 5	57.02	42.29	48.56
Top 4	47.10	50.18	48.60
Top 3	54.27	46.59	50.14
Top 2	52.16	48.71	50.37
Top 1	56.83	**55.57**	56.19
BLSTM	69.79	41.31	51.90
BLSTM-CRF	**70.15**	41.06	51.80
BLSTM-MLC	68.03	48.53	56.65
JD	67.61	48.97	56.80
JT+JD	65.44	50.53	**57.03***

Table 5.5: Performance of event detection with respect to types (span+type). The star (*) indicate statistical significance compared to the BLSTM-MLC model at $p < 0.05$, based on a two-tailed paired t-test.

Model	Precision	Recall	F1
Top 5	35.01	32.70	33.81
Top 4	43.22	32.05	36.81
Top 3	39.69	38.81	39.24
Top 2	42.52	36.50	39.28
Top 1	38.51	**41.03**	39.73
BLSTM	55.09	32.61	40.97
BLSTM-CRF	**55.20**	32.31	40.76
BLSTM-MLC	52.84	37.69	44.00
JD	52.56	38.07	44.16
JT+JD	50.72	39.16	**44.20***

Table 5.6: Overall performance of event detection (span+type+realis). The star (*) indicate statistical significance compared to the BLSTM-MLC model at $p < 0.05$, based on a two-tailed paired t-test.

Table 5.7 shows the result of event coreference resolution in the TAC KBP 2017 dataset. As shown, our joint modeling approaches also improve the performance of event coreference resolution. The performance difference between JT+JD and NEC is statistical significant at $p < 0.05$, based on a two-tailed paired t-test.

We provide an analysis of the JT+JD model, giving some examples in the TAC KBP test data. We first show how the JT+JD model works successfully as compared to the BLSTM-MLC model. In Example (78) below, both BLSTM-MLC and JT+JD can recognize E120 correctly, including its span and type. However, BLSTM-MLC fails to detect E121 as an event, presumably because the context surrounding the event nugget is quite different from those observed in training data, ending up with a low probability. The event detection subnetwork in JT+JD also finds that E121 is unlikely to be an event of 'Elect' with a probability of 0.35, but the joint processing of JT+JD allows the model to detect the event nugget through event coreference between E120 and E121. Similarly, both BLSTM-MLC and JT+JD can recognize E120 in Example (79) correctly. However, only JT+JD correctly detects E121 as an event of 'Transport-Person' via event coreference between E120 and E121. These examples illustrate how the joint model can make more informed decisions through event coreference, thereby improving the performance of both event detection and event coreference resolution.

Model	MUC	B^3	$CEAF_e$	BLANC	CoNLL Avg	TAC KBP Avg
Top 5	12.57	24.98	23.36	8.96	20.30	17.47
Top 4	19.30	28.66	28.64	13.56	25.53	22.54
Top 3	22.90	34.34	33.63	17.94	30.29	27.20
Top 2	33.79	39.88	35.73	26.06	36.47	33.87
Top 1	30.63	43.84	39.86	26.97	38.11	35.33
LTR (Baseline)	29.94	43.92	41.60	25.64	38.49	35.28
NEC-TR	30.19	44.38	42.88	26.17	39.15	35.91
NEC	33.95	44.88	43.02	28.06	40.62	37.48
JD	34.04	**45.02**	**43.15**	28.15	40.74	37.59
JT+JD	**35.81**	44.87	41.98	**29.47**	**40.89***	**38.03***

Table 5.7: Performance (F1 scores) of event coreference resolution in the TAC KBP 2017 dataset. 'Top N' represents the Nth-ranked system reported in the official results. The stars (*) indicate statistical significance compared to the NEC model at $p < 0.05$, based on a two-tailed paired t-test.

(78) For those not in Ontario, there's a provincial **election**(E120) next week, quite hotly contested at the moment. ... I thought that was going to be a "bureaucracy" dream until you got to the **election**(E121) part and I was going to ask if you had been a civil servant.
Elect Elect

(79) I was talking to my 26 yr youngest daughter yesterday and it appears she is considering upping sticks and **moving**(E122) back to the UK. Actually that is just the excuse she wants as she has expressed the desire to **move**(E123) back for a while now but it appears, a Trump presidency would the last straw.
Transport-Person Transport-Person

On the other hand, we also observed that the issues of pronouns and abstract words described in Section 5.1 are still challenging to JT+JD. In Example (80), 'it' in the second sentence corefers to 'leave' in the first sentence, and the JT+JD model detects 'leave' as a correct event nugget of 'Transport-Person'. However, the probability of 'it' being an event nugget of 'Transport-Person' is very low (approximately 5.8×10^{-5}) and much less than the optimal value of p_c. Thus, JT+JD does not consider 'it' in the joint process with event coreference resolution. We also observed the same issue in the case of the abstract word 'event' in Example (81). JT+JD detects 'protest' correctly as an event nugget of 'Demonstrate', but the probability of 'event' being an event nugget of 'Demonstrate' is 3.1×10^{-8}. In the case of closed-domain event tasks such as TAC KBP, pronouns and abstract words are rarely in-domain event nuggets, and a large number of negative examples make it difficult to detect them correctly. We leave this challenge for future work.

(80) Why would you **leave**(E124)? If certain things changes, then maybe that would be a time to think about *it*, but an election may change who is in power.
Transport-Person

(81) Aaron Black, the **protest**(E125)'s organizer, said the *event* is meant to shed light on the NRAs part in hindering progress for improving gun legislation.
Demonstrate

5.4 Related Work

There is almost no prior work on joint models that simultaneously perform event detection and event coreference resolution. Thus, in this section we provide a literature review of joint structured learning models in other NLP tasks. We first focus on the joint structured learning of feature-based models in NLP (Section 5.4.1), and then move on to the joint structured learning of neural network models in NLP (Section 5.4.2).

5.4.1 Joint Learning of Feature-based Models in NLP

The underlying idea of joint structured learning is that one can train a structured learning model to globally capture the interactions between two relevant tasks via a certain kind of structure, while making predictions specifically for these respective tasks. Feature-based models make this joint modeling feasible by featurizing and scoring each part of the output structure. Joint learning of feature-based models has been studied in several pairs of relevant NLP tasks. Many joint models leverage an incremental token-level structured perceptron in combination with beam search for joint decoding. Zhang and Clark (2008a) is the first work to propose the multiple-beam search algorithm, and present the joint structured perceptron approach for Chinese word segmentation and part-of-speech (POS) tagging. Their system successfully achieves error reduction in the both tasks. This technique has been further extended to capture some specific graph structure for parsing. (Li et al., 2013) apply the joint structured perceptron algorithm to the extraction of event triggers and arguments, and Li and Ji (2014) apply to the extraction of entity mentions and relations, both formalizing the problem as graph structure learning. In a similar vein, Johansson and Nugues (2008) propose an online structured learning algorithm for jointly performing dependency parsing and semantic role labeling. Bohnet and Nivre (2012) show that it is feasible to implement a joint decoding mechanism in a transition-based system for joint POS tagging and dependency parsing.

5.4.2 Joint Learning of Neural Network Models in NLP

Joint learning of neural network models is much less studied, as compared to a host of the prior work on feature-based models described in Section 5.4.1. Language modeling and translation modeling have been explored to leverage both source and target words (Auli et al., 2013; Devlin et al., 2014). Nguyen et al. (2016) use a bidirectional Gated Recurrent Unit (GRU) model that leverages memory vectors and matrices to store the prediction information and predict arguments jointly with triggers.

5.5 Summary

In this chapter, we first focused on the phenomenon of event interdependencies and the problem of error propagation in pipeline models, stemming from performing event detection and event coreference resolution separately. We first proposed a joint feature-based structured prediction model for event trigger identification and event coreference resolution. Our experiment shows

that the proposed method effectively penalizes false positives in joint search, thereby outperforming a pipelined model substantially in event coreference resolution. We also proposed a joint neural model for event detection and event coreference resolution on the TAC KBP dataset, with two techniques of joint decoding and joint training. Our experiment demonstrated that the joint model outperforms state-of-the-art models (top systems reported in the official results) and our strong pipelined baseline in both event detection and event coreference resolution.

Chapter 6

Applications of Events

The semantic and discourse structures of events have been utilized in a wide variety of NLP applications, as described in Section 1.1. In this chapter, we discuss how events and their coreferences can be helpful to the applications. In particular, we present our own application for question generation in Section 6.1. We also provide a literature review of other applications of events (Section 6.2). The work described in Section 6.1 is based on (Araki et al., 2016).

6.1 Question Generation

As described in Section 1.4.5, most applications of event coreference have let a target system use matched pairs of coreferent event mentions for a downstream task, such as information extraction (Humphreys et al., 1997), topic detection and tracking (Allan, 2002), textual entailment (Haghighi et al., 2005), or contradiction detection (de Marneffe et al., 2008). However, what is not well studied is applications which involves *humans* in resolving event coreferences. In this section, we present a novel approach to question generation as an example of the applications to facilitate humans' semantic understanding of texts.

Question generation is the task of generating questions from plain text automatically, often studied from an educational perspective. This is because exam questions are an indispensable tool for teachers to assess their students' understanding of material, and question generation systems can assist teachers to create such questions. Thus, automatic question generation from text is a key NLP technology to examine learners' reading comprehension. Past studies in education show that higher-level questions, in contrast to simple factoid questions, have more educational benefits for reading comprehension (Anderson and Biddle, 1975; Andre, 1979; Hamaker, 1986). However, most of existing approaches have focused on generating questions from a single sentence (Mitkov and Ha, 2003; Chen et al., 2009a; Heilman and Smith, 2010a; Curto et al., 2011; Becker et al., 2012; Lindberg et al., 2013; Mazidi and Nielsen, 2014), relying heavily on syntax and shallow semantics with an emphasis on grammaticality. A problem with this approach is that the majority of questions generated from single sentences tend to be too specific and low-level to properly measure learners' understandings of the overall contents of text. In other words, what is assessed by such question generation systems ends up essentially being the ability to compare sentences, just requiring learners to find a single sentence that has almost the same surface form

as a given interrogative sentence. Results of simple sentence comparisons do little to contribute towards the goal of assessing learners' reading comprehension.

To address the issue, we propose an approach that engages learners through the use of specific *inference steps* over multiple sentences, namely coreference resolution and paraphrase detection, requiring more semantic understanding of text. We primarily use event and entity coreference as a source of producing questions from multiple sentences. Grounded by the past studies in education, we believe that such high-level questions are more sophisticated and educationally valuable for testing reading comprehension than questions generated by the traditional single-sentence approach. Our question generation strategy is novel in two ways. First, it employs event and entity coreference between antecedents and referring mentions spanning multiple sentences. This requires learners to resolve the coreference and understand the contents of the text semantically. Second, it makes use of paraphrases when generating questions. The resulting questions are able to check learners' lexical knowledge.

6.1.1 Generating Questions using Coreferences and Paraphrases

Our question generation system is aimed at enhancing the reading comprehension ability of language learners, more specifically, students who learn English as a second language (ESL). Therefore, our ultimate goal is to generate multiple-choice questions from plain texts in an arbitrary domain. However, the state-of-the-art technology for extracting semantic representations of event and entity relations from text does not perform well enough to support our question generation approach. Thus, the evaluation of question generation relying on automated semantic relation extraction is not practical at this time. In this work, therefore, we use texts and expert human annotations from the ProcessBank corpus[1] to facilitate our semantics-oriented question generation. The ProcessBank corpus comprises 200 paragraphs about biological processes, extracted from a high school level textbook. Such textbooks are ideal sources to test learners' reading comprehension from an educational perspective. In addition to the expert annotation of events and entities, the corpus also includes multiple-choice questions per paragraph, created by the biologists. We refer to these expert questions to devise our question templates.

Our question generation strategy is primarily aimed at generating questions from multiple sentences using three semantic relations: event coreference, entity coreference, and paraphrases. We recognize that one of the key learning points for biology students is the order of biological processes (events) because many of the expert questions in ProcessBank ask about it. Based on this observation, we devise question patterns and templates focusing on biological processes and their order, shown in Table 6.1 on page 103.

Our question generation system consists of two components: **answer generation** and **question construction**. Figure 6.1 on page 104 gives an example of a paragraph in ProcessBank to illustrate how the question generation system works. An example question generated from the event coreference between "divide" and "division" using template T5 is: "What is a result of the fibroblast division not only in the artificial conditions of cell culture, but also in an animal's body?" First, the answer-generation component finds question patterns applicable to the given text. For instance, pattern P3 is applicable to the paragraph of Figure 6.1 since "divide"

[1] See Section 2.1.4 for details of the corpus.

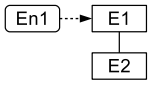
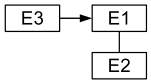
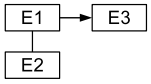
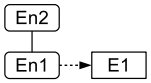
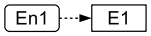
Semantic relation	Question patterns	Answer	Question templates
Event coreference	P1. 	En1	T1. What [*verbal trigger* + *subsequent arguments*]?
	P2. 	E3	T2. What causes [*nominal trigger* + *subsequent arguments*]? T3. What makes it happen to [*verbal trigger* + *subsequent arguments*]? T4. What makes it happen that [*event clause*]?
	P3. 	E3	T5. What is a result of [*nominal trigger* + *subsequent arguments*]? T6. What happens when [*event clause*]?
Entity coreference	P4. 	En2	T1. What [*verbal event trigger* + *subsequent arguments*]?
Paraphrase	P5. 	En1	

Table 6.1: Question patterns and templates using event coreference, entity coreference, and paraphrases. In question patterns, En denotes an event trigger, and Enn an entity mention. A straight line denotes a coreference link, a dashed arrow an 'Agent' relation, and a straight arrow a relation which is 'Cause', 'Enable' or 'Result'. An event clause in question templates is defined as a text span including an event trigger and its arguments.

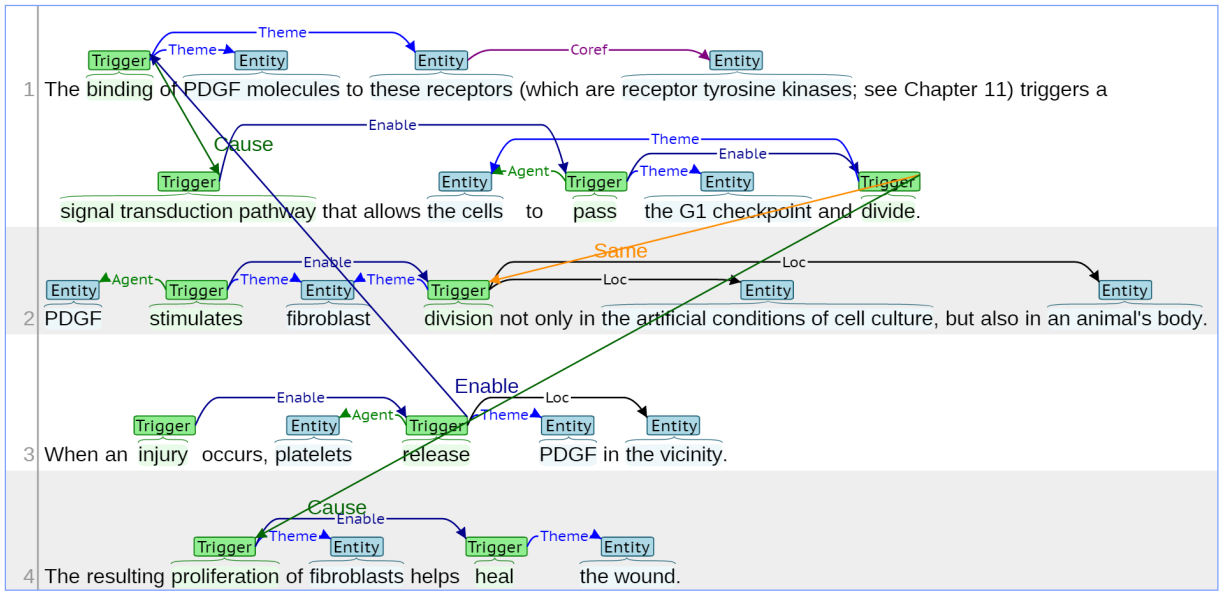


Figure 6.1: A paragraph with annotation of events, entities and their relations in ProcessBank. A 'Same' link means event coreference, whereas a 'Coref' link means entity coreference.

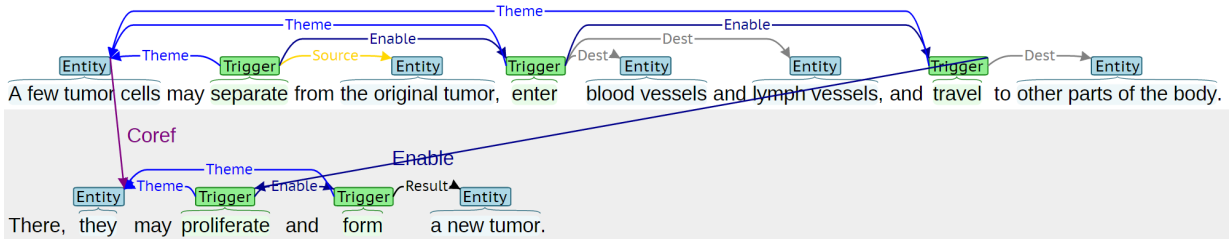


Figure 6.2: An example text to generate a question using an entity coreference.

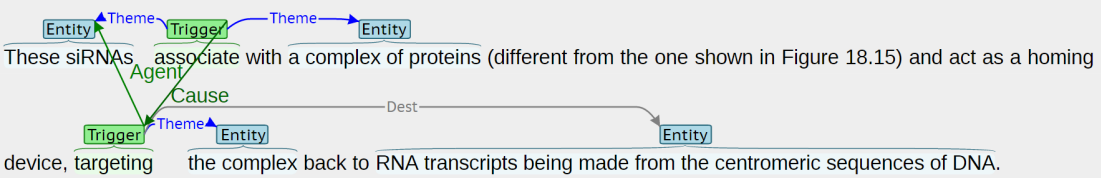


Figure 6.3: An example text to generate a question using a paraphrase.

(corresponding to E1 in P3) in the first sentence and "division" (corresponding to E2 in P3) in the second sentence are coreferent, and only the former trigger has a 'Cause' relation to "proliferation" (corresponding to E3 in P3). This pattern match means that "proliferation" can be an answer. We then make use of arguments of the answer trigger to generate a more complete answer, obtaining "proliferation of fibroblasts". Second, the question-construction component creates a question given the matched pattern. In the case of the example above, "division" is a nominal trigger, and thus the algorithm uses question template T5, creating the question above. As shown in this example, the algorithm takes advantage of the fact that E2 lacks a piece of information that E1 has in P1, P2 and P3. We only use event coreference E1-E2 where E1 and E2 exist in different sentences, ensuring that questions are generated from multiple sentences.

We also give examples to illustrate how to generate questions based on entity coreferences and paraphrases in Figure 6.2 and Figure 6.3 on page 105, respectively. An example question generated from the text in Figure 6.2 is: "What may proliferate and form a new tumor?" Pattern P4 applies to the text because "they" (En1) in the second sentence is coreferent with "a few tumor cells" (En2) in the first sentence, and the former entity is an agent of triggers "proliferate" and "form". Given that, "a few tumor cells" can be an answer, and the system can generate the question above. Another example question generated from the text in Figure 6.3 is: "What targets the composite back to RNA transcripts being made from the centromeric sequences of DNA?" P5 applies to this case because "these siRNAs" are an agent of trigger "targeting", and "complex" has a paraphrase "composite". Since "these siRNAs" are coreferent with "siRNAs" in the previous sentence (not appearing in Figure 6.3), a system can generate an answer "siRNAs" and the question above. Note that the paraphrase "composite" is inserted into the question.

6.1.2 Evaluation of Generated Questions

It is important to evaluate generated questions, but this is not straightforward mainly due to the wide variety of acceptable natural language expressions. We define three metrics to measure the performance of a question generation system, as shown below.

Grammatical correctness judges whether a question is syntactically well-formed. It does not evaluate whether a question is semantically coherent, ignoring the meaning of the question. Our three point scale for this metric is based on the number of grammatical errors.

- 1 (best): The question has no grammatical error.
- 2: The question has 1 or 2 grammatical errors.
- 3 (worst): The question has 3 or more grammatical errors.

For the consistency in counting grammatical errors, we define common grammatical errors in English: spelling errors, run-on sentences, lack of subject-verb agreement, lack of pronoun-antecedent agreement, misplaced modifiers, missing or erroneous quantifiers, prepositions or

determiners, erroneous verb forms or nominalization, incorrect word choice, and other errors.

Answer existence identifies whether the answer to a question can be inferred from the passage associated with the question. Note that the answer must be inferred using the passage information only, without relying on external knowledge beyond the passage. Even if a system generates a question while making a specific target its answer, it could be impossible that the target is the answer due to the lack of a valid inference path from the question to the target as its answer. This metric is intended to penalize such questions. Our two-point scale for this metric is as follows:

- 1 (yes): The answer to the question can be inferred from the passage.
- 2 (no): The answer to the question cannot be inferred from the passage.

In addition to answer existence, we also evaluate the correctness of system-generated answers. For this, we use the following three-point scale ratings: correct (1), partially correct (2), and incorrect (3).

Inference steps concern how many semantic relations humans need to understand in order to answer a question. This metric directly evaluates our central idea: inference steps for answering a question. We define the following set of semantic relation types to be considered as inference:

- Event coreference within input text and event coreference between input text and a question.
- Entity coreference within input text and entity coreference between input text and a question.
- Paraphrases in input text and a question.
- Negation which is a binary relation about logical truthness.

6.1.3 Experiments and Discussions

To assess the performance of our system, two human annotators evaluate our question generation component and distractor generation component. For a meaningful comparison on question generation, we use the question generation system by Heilman and Smith (2010a) as a baseline. Let MH refer to the baseline. In our experiments, we generate 200 questions from each system.

We show our results of question generation in Table 6.2. Our question generation algorithm achieved more inference steps compared to MH by 0.60, while it gained comparable ratings of grammatical correctness and answer existence. We computed the inter-annotator agreement with Cohen's Kappa for each of the criteria mentioned in Section 6.1.2. Overall, we have a kappa value of 0.55, 0.58 and 0.49 for grammatical correctness, answer existence and inference steps respectively. This result implies moderate agreement. Table 6.3 shows our results of answer correctness. We observe that our question generation algorithm tends to generate questions with more incorrect answers than MH.

As described in Section 6.1.1, our question generation approach attempts to generate questions involving at least one inference step. The average inference step of 0.76 in Table 6.2 means that the algorithm fails to generate intended questions approximately once out of every four times. A common source of the errors happens in questions generated using event coreference; some other events can be associated with the event in the question via a relation other event coreference (e.g., super-event), and they can be an answer to the question. As for grammatical correctness, the algorithm made errors on determiners in the case of question patterns involving

System	Grammatical correctness			Answer existence			Inference steps		
	Ann 1	Ann 2	Total	Ann 1	Ann 2	Total	Ann 1	Ann 2	Total
Proposed	1.52	1.48	1.50	1.17	1.26	1.21	0.80	0.71	**0.76**
MH	1.42	1.25	**1.34**	1.20	1.14	**1.17**	0.13	0.19	0.16

Table 6.2: The performance comparison in question generation. Numbers in grammatical correctness and answer existence are average ratings, and lower is better. Numbers in inference steps are average inference steps, and higher is better.

System	Ann 1	Ann 2	Total
Proposed	1.35	1.57	1.46
MH	1.08	1.13	1.11

Table 6.3: Average ratings of answer correctness in 200 questions. Lower numbers are better. Scores range from 1-3, with 1 a correct answer.

a nominal trigger. For instance, "natural selection" is a noun phrase which does not need an article, but it mistakenly adds "the" in front of it.

6.2 Related Work

The semantic argument and discourse structures of events have been already utilized in a wide range of NLP applications, as described in Section 1.1. In this section, we provide a more detailed literature review of these applications, particularly paying attention to how the event structures have been utilized in the applications.

Text summarization is the task of automatically condensing a document text into a shorter summary that retains the most important concepts mentioned in the input text. Extractive summarization is a type of text summarization whose goal is to construct a summary by extracting sentences which contain the most salient concepts in text. Several researchers have explored event-based extractive summarization by employing argument structures of events (Filatova and Hatzivassiloglou, 2004; Li et al., 2006; Liu et al., 2007; Zhang et al., 2010). The event-based extractive summarization makes an underlying assumption that the argument structures of events can capture important information about entities, events and their relations, thereby being able to represent the significance of sentences. A traditional approach is to construct a co-occurrence graph of named entities and (verbal) event triggers by leveraging lexical resources such as WordNet (Miller et al., 1990) and VerbOcean (Chklovski and Pantel, 2004), and rank sentences or triggers with respect to their significance computed by the PageRank algorithm or a clustering algorithm. More recent studies incorporate temporal information (Gung and Kalita, 2012) and distributed word representations (Marujo et al., 2016) into event-based summarization.

Knowledge-base population (KBP) is the task of gathering information from a large text corpus to complete deficient elements of a knowledge base. The task has been done through two sub-tasks: (1) entity linking whose goal is to link entity mentions in text to entities in the knowledge base, and (2) slot filling whose goal is to complete all known information about a given entity (McNamee and Dang, 2009). One example of entity linking is to resolve entity mention

"Obama" in text to "Barack Obama" in the knowledge base, and one example of slot filling is to collect information regarding Barack Obama such as his birthplace, birthdate, occupation, and so forth. Such information is often given in the form of events (i.e., who did what to whom where and when), and therefore event extraction technologies lend themselves to slot filling.

Information extraction (IE) is the task of extracting specified types of information from a natural language text. Humphreys et al. (1997) propose a rule-based approach where they use event coreference to put constraints on the template-filling process in IE, and show a performance gain in the management succession task of the sixth Message Understanding Coreference (MUC-6) (Sundheim, 1995). The argument structure of events is a core representation used in IE, as described in Section 1.1. Therefore, the outcome of event extraction refined by resolved event coreferences directly gives an integral basis to IE.

Topic detection and tracking (TDT) is aimed at searching, organizing and structuring multilingual, news oriented textual materials from a variety of broadcast news media (Allan, 2002). In TDT, a topic is defined to be a set of news stories that are strongly related by a certain seminal real-world event. TDT comprises five major tasks:

1. *Story segmentation*: dividing the transcript of a news show into individual stories.
2. *First story detection*: recognizing the onset of a new topic in the stream of news stories.
3. *Cluster detection*: grouping all stories that discuss the same topic.
4. *Topic tracking*: monitoring news stories similar to a set of example stories.
5. *Link detection*: deciding whether two stories discuss the same topic (topically linked).

The fifth task is analogous to event coreference resolution, but it differs in that the equivalence between two document-level stories is questioned, rather than clause-level event mentions studied in this thesis. Glavaš and Šnajder (2013b) make use of a cross-document event coreference resolution model by Glavaš and Šnajder (2013a) to compute a similarity between two news stories via event graph kernels.

Paraphrase extraction is the task of finding sets of paraphrase texts from a given text collection. The task is strongly related to cross-document event coreference resolution in the sense that the both tasks are tackled with similar techniques, as shown in (Bagga and Baldwin, 1999; Tomadaki and Salway, 2005; Regneri and Wang, 2012). As implied from the task resemblance, paraphrase extraction can benefit from the outcome of an event coreference resolution system. For instance, if we collect a set of resolved event coreference pairs with different surface forms of triggers or nuggets, it will be a valuable set of candidates for word-level or phrase-level paraphrases.

Textual entailment is the task of automatically determining whether a natural language hypothesis can be fully inferred from a given piece of natural language text, as illustrated in the following example:

- Text: Amazon was founded by Jeff Bezos in 1994.
- Hypothesis: Bezos established a company.

A popular strategy in previous studies on textual entailment is to approximate entailment by means of a certain semantic similarity score between the text and the hypothesis. For the similarity computation, they often explore a kind of structural sentence similarity to capture the semantic equivalence between the text and the hypothesis. This computation resembles the com-

putation of probabilities or scores of event coreference in a pairwise model. For computing the structural sentence similarity, Haghighi et al. (2005) present an approach to leverage dependency graph matching. Zanzotto and Moschitti (2006) employ tree kernels by (Collins and Duffy, 2002) to augment syntactic trees with placeholders. Mehdad et al. (2010) extend the tree-kernel approach by incorporating lexical semantic information from WordNet and via distributed semantics through Wikipedia. More recently, Rocktäschel et al. (2016) and Liu et al. (2016a) explore sentence-level distributed representations in a Long Short-Term Memory (LSTM) model with attention mechanisms.

Contradiction detection is the task of automatically finding contradictions in text. A standard definition of contradiction is that sentences A and B are contradictory if there is no possible world in which A and B are both true. In this task, event coreference can be assumed as a necessary condition for contradiction (de Marneffe et al., 2008). Namely, events described in two sentences need to be coreferent in order for the sentences to be contradictory. Otherwise, the two sentences cannot be contradictory to each other. One example pair of contradictory sentences is as follows:

(82) President Kennedy was assassinated in Texas.

(83) Kennedy's murder occurred in Washington.

The two sentences refer to the same event, and the location mismatch render them contradictory. Using event coreference, one can filter out unrelated sentences with non-coreferent events to avoid false positives of contradiction.

Lastly, **stock market prediction** is the task of automatically forecasting stock price movements, and is of clear interest to financial institutions and individual investors. One of the main factors that move the stock price of a particular company is company-related events described in financial news. This is because many of such events are analyzed with respect to the company either positively or negatively, resulting in a positive or negative impact on its stock price. Ding et al. (2014) formalize a structured representation of an event by a quadruple (O_1, P, O_2, T) where P is an action, O_1 is an actor, O_2 is an object, and T is a timestamp. To extract events, they detect predicate verbs and their arguments using the Open IE technologies by Fader et al. (2011) and generalize them to events using WordNet and VerbNet. They present two models for stock price prediction using Support Vector Machines (SVMs) and a deep neural network model, and show that event information improves upon both models. Lee et al. (2014) suggest that the use of event-related features such as event categories in the form 8-K are a good indicator for their feature-based classifier using random forests.

6.3 Summary

In this chapter, we reviewed the problem of limited applications of events stated in Section 1.4.5, and presented our novel approach to question generation (QG) that utilize events and event coreferences in order to address the problem.

As for QG, we showed how events and event coreferences can contribute to more sophisticated question generation from multiple sentences, as compared to the traditional approach to question generation from single sentences. Our QG system automatically generates questions

which require learners to have semantic understanding of text by taking specific inference steps over multiple sentences. From our experiments, we observed that questions generated by our approach require a larger number of inference steps while ensuring comparable grammatical correctness and answer existence, as compared to the traditional single-sentence approach.

In addition to our application to QG, we provided a literature review of other applications of events. In particular, we discussed how the applications in previous studies employed the argument structures of events and event coreferences.

Chapter 7

Conclusion

In this thesis, we first described five problems with state-of-the-art approaches to events and their coreferences in Section 1.4, and proposed novel solutions to each of the problems. Below is the conclusion of this thesis with respect to the goal and contributions described in Section 1.5:

1. **Open-domain event detection** (Chapter 3). We introduced a new paradigm of open-domain event detection to overcome issues of prior work on restricted domains and syntactic types. Based on our annotation guidelines (Appendix A), we manually annotated event nuggets in Simple Wikipedia articles in 10 different domains such as geology and economics. The annotated events comprise verbs, nouns, adjectives, and phrases which are continuous or discontinuous (Section 3.2). Despite this relatively wide and flexible annotation of events, we achieved high inter-annotator agreement: 80.7% F1 (strict match) and 90.3% F1 (partial match). To facilitate future studies on event detection, we release the new corpus of human-annotated events.
2. **Distantly-supervised methods for open-domain event detection** (Chapter 3). Due to the ubiquity and ambiguities of events, human annotation of events in the open domain is substantially expensive. We proposed a distant supervision approach to open-domain event detection, thereby circumventing the data sparsity problem. Our distant supervision method is able to generate high-quality training data automatically, obviating the need for human annotation Section 3.5. The method is not bounded to any particular datasets and offers a versatile solution for event detection. Our experiment shows that the model outperforms supervised models in both in-domain and out-domain settings.
3. **Subevent structure detection** (Chapter 4). We presented a novel two-stage approach for finding and improving subevent structures (Section 4.2). This is the first work that computationally detects subevent parent-child relations. The first stage employs multiclass logistic regression to identify subevent parent-child and subevent sister relations. The second stage selects out parents for detected subevents by a probabilistic voting algorithm. Our experimental results show that the first stage achieves reasonable performance, and the second stage improves the performance of subevent detection. We also proposed an evaluation scheme for partial event coreference resolution by introducing the notion of conceptual event hierarchy (Section 2.2.3). We examined three link-based metrics and showed that extensions to MUC and BLANC are more adequate than an extension to the

simple tree match algorithm.

4. **Joint modeling for event detection and event coreference resolution** (Chapter 5). To address the problem of error propagation in pipeline models, we proposed two joint models that can simultaneously identify events and resolve event coreferences using structured perceptron (Section 5.2). First, our feature-based structured prediction model leverages an incremental decoding algorithm that combines the segment-based decoding and best-first clustering algorithm. This algorithm avoids a problem that the incremental token-based prediction in joint decoding poses a challenge of synchronizing the assignments of event triggers and coreference. Our experiment shows that the proposed method effectively penalizes false positives in joint search, thereby outperforming a pipelined model substantially in event coreference resolution. Second, we also proposed a joint neural model for event detection and event coreference resolution, with two techniques of joint decoding and joint training. Our experiment demonstrated that the joint model outperforms state-of-the-art models (top systems reported in the official results) and our strong pipelined baseline in both event detection and event coreference resolution (Section 5.3).
5. **Applications of events** (Chapter 6). We presented a novel applications of events for question generation (QG). We showed how events and event coreferences can contribute to more sophisticated question generation, aiming at human learners through the use of specific inference steps over multiple sentences (Section 6.1). Our experiments indicate that questions generated by our approach require a substantially larger number of inference steps while ensuring comparable grammatical correctness and answer existence, thereby necessitating deeper semantic understanding of texts. This application illustrates the importance of event structures in natural language understanding by humans.

7.1 Future Work

For future work, one can address many more research problems with event structures. We highlight some of the immediate future directions below.

- **Incorporating external knowledge into event coreference resolution**. In this thesis, we study a method that incorporates linguistic and domain knowledge into a neural network model for event detection. The motivation of this work is based on the definition of event nuggets, which are a semantically meaningful event expressions. With a similar motivation, one could study a method that incorporates external knowledge for resolving event hoppers. This is because event hoppers allow semantically lenient argument match (e.g., "Thursday" vs. "last week"), and external knowledge can be also helpful to the decision of such argument match.
- **Semi-supervised learning for event coreference resolution**. We explore a semi-supervised learning approach to event detection in this thesis. Similarly to event detection, a dearth of training data is a serious problem in event coreference resolution as well since human annotation of event coreference is also expensive. One could investigate semi-supervised learning models to mitigate the issue of data sparsity in event coreference resolution.
- **Cross-document event coreference resolution**. In this thesis, we have worked on within-

document event coreference resolution. That is, we assume that every event coreference occurs in a single document. In addition to the within-document event coreference, one could study cross-document event coreference where two event mentions in different documents refer to the same event.

- **Multilingual event detection and event coreference resolution**. This thesis focuses on event detection and event coreference resolution for text written in English. However, events are language-independent phenomena, and thus event structures are prevalent in any natural language other than English. Thus the problems addressed in this thesis can be tackled in other languages.

Appendix A

Annotation Guidelines for Open-Domain Event Nuggets

This appendix provides our guidelines for annotating events in the open domain.

A.1 Introduction

Our event annotation is based on the notion of eventualities (Bach, 1986) and event nuggets (Mitamura et al., 2015b). In this annotation, we focus on event spans and ignore other attributes such as event types and realis. We first introduce the following notations to clarify event annotation:

- **Boldface** means that marked word or phrase is highlighted as eventive;
- *Italic face* means that marked word or phrase is highlighted as non-eventive;
- An underline means that marked words are grouped together as a single unit.

A.2 Principles of Event Annotation

We approach annotation of events from two perspectives: semantic perspective (Section A.2.1) and syntactic perspective (Section A.2.2).

A.2.1 Semantic Perspective: Eventualities

Our definition of events is based on eventualities, which are a broader notion of events and consist of actions, processes, and states. More specifically, events are verbs, nouns, adjectives and phrases that refer to:

1. actions that involve a change of state with an explicit goal or completion, e.g., **walked** to Boston, **buy** a book;
2. processes that involve a change of state without an explicit goal or completion, e.g., it was **raining** yesterday; and
3. states that remain unchanged until their change or are brought as a result of an event, e.g., He **owns** a car. Tom was **happy** when he **received** a present. ("happy" is a state implying

that Tom's reception of the present made him happy; see Section A.6 about annotating adjectives; "received" is an action).

Note that we introduce the notion of eventualities in order to clarify the semantic boundary between eventives and non-eventives, not because we are interested in differentiating the three classes above. Annotating states is generally more difficult than annotating actions and processes because states are often confused with attributes which are not eventive. Our basic policy is that we annotate states if they imply actual occurrences. For more details, see Section A.4, Section A.5, Section A.6 and Section A.7.

A.2.2 Syntactic Perspective: Event Nuggets

We also define what textual units are annotated as events. For this purpose, we use the notion of event nugget, which is defined as a semantically meaningful unit that expresses an event. An event nugget can be either a single word (verb, noun, or adjective) or a phrase which is continuous or discontinuous, depending on how we interpret the semantic meaningfulness of an event that the event nugget refers to. Below are examples of event nuggets:

- He **shot** the teller in the bank.
- He **opened fire** at the teller in the bank.
- He **turned** the television **on**.

In the first example, "shot" is the only verb representing an event, and we annotate "shot" as a single-word event nugget. On the other hand, we annotate "open fire" as a continuous multi-word event nugget in the second example because the phrase "open fire" is a more semantically adequate unit to express the corresponding event than either "opened" or "fire." Similarly, we annotate "turned ... on" as a discontinuous multi-word event nugget in the third example.

A.3 General Rules

In the subsequent sections, we guide how to annotate or not to annotate verbs, nouns, adjectives, and adverbs. This section describes general rules for expressions besides the four syntactic types.

(1) Do not annotate articles (e.g., a, an, the).
- The company **filed** a **lawsuit**. (Only annotate "file ... lawsuit" as a discontinuous event nugget; do not include "a" before "lawsuit")
- After the **shooting**, many people ... (Don't annotate "the" before "shooting")

(2) Do not annotate prepositions that may look like verbs but implies no actions (e.g., including, like).
- Six people were **killed** in the riot, *including* a policeman. (Don't annotate "including" because it is a preposition, not a verb, which is semantically similar to "such as")
- She's **wearing** a dress *like* mine. (Don't annotate "like" it is a preposition meaning "similar to," which implies no actions)

(3) Do not annotate words indicating negation (e.g., not, never, no, neither).
- Tom did *not* **eat** lunch yesterday.
- *No* pilots could **see** each other.
- The ship *never* **returned**.

- She *never* **mishandled** the equipment. (Annotate "mishandled" even though 'mis-' is a prefix indicating negation)

(4) Hyphenation
When multiple words are connected by one or more hyphens to indicate a particular meaning, we annotate the entire part of connected words as a single event.
- There is a **man-made** river in the country.
- It is an **often-cited** project.
- **Well-known** researchers **gave a speech** at the **conference**. ("gave ... speech" is a single event nugget; see the description about annotating light verbs in Section A.4)

A.4 Annotating Eventive Verbs

This section provides guidelines about how to annotate eventive verbs and what verbs not to annotate. We first describe how to annotate eventive verbs. We consider most verbs as events. It is generally straightforward to annotate normal verbs (verbs expressing physical actions, e.g., walk, run, eat, shoot), but there are sometimes difficulties determining the eventiveness of some types of verbs. We enumerate such types of verbs and describe whether they are annotated or not, giving some examples. We also describe how to annotate verbal phrases (e.g., look for, carry out, etc). In order to decide a verbal phrase, we suggest examining contexts and looking up a dictionary such as WordNet, Oxford or Wiktionary. In addition, we explain the difference between verbal phrases we annotate and the ones we don't.

(1) Psychological or cognitive verbs are verbs concerned with mental cognition. Examples are: see, know, hear, feel, find, believe, think, estimate, decide, consider, understand, misunderstand, acknowledge, perceive, expect, etc. We annotate all psychological/cognitive verbs.
- I **believe** it is true.
- I **found** it interesting.
- It is now **known** that they were bullies.
- They **figured out** the problem. (Annotate "figured out" together as a single event nugget because it is a phrasal verb meaning "understood")

(2) Aspectual verbs are verbs specifying aspects, i.e., temporal properties of actions. Examples are: begin, stop, start, continue, finish, last, remain, etc. We annotate all aspectual verbs.
- It **remained** an independent country for ten years.
- The ice has **begun** to **melt**. ("begun" is an aspectual verb that we annotate; "melt" is also an event)

(3) Causative verbs are verbs indicating that someone or something makes something happen. Examples are: cause, make, let, lead, help, result, have, end up, force, prevent, etc.
- He **forced** me to **cancel** the **flight**. ("forced" is a causative verb denoting an event; "cancel" and "flight" are two independent events)
- A large **earthquake** can **cause** a **tsunami**. ("cause" is a causative verb)
- The poor **weather** may have **accounted for** the small crowd. (Annotate "accounted for" together as a single event nugget because it is a phrasal verb meaning "explain")
- She **made** me **happy**. (Annotate "made ... happy" together as a single nugget because it is a semantically meaningful unit)

(4) Performance verbs are verbs meaning to do or not to do something. Examples are: perform, carry out, conduct, do, fail, etc.
- He **conducted** three experiments.
- She **failed** to **come back** by 7 o'clock. ("failed" is a performance verb, and note that "come back" is a phrasal verb referring to another event)
- We had to **solve** the problem. So we **did it**. (Annotate "did it" together as a single unit because it is a semantically meaningful unit referring to "solving the problem")

(5) Bridging or supporting verbs are verbs that come before main verbs and provide additional meanings for the main verbs. These verbs and their subsequent verbs showing full semantic contents are annotated separately as event nuggets. Examples are: try, help, allow, enable, etc.
- He **tried** to **persuade** his parents. ("tried" indicates that he puts his effort when he was persuading his parents)
- His **support helped** me (to) **complete** the project. ("helped" indicates that his support led to the completion of the project)

(6) Communication verbs are verbs that express some kind of communication between entities. Examples are: say, describe, call, name, tell, speak, etc.
- We **call** our dog Jack.
- He was **named** after his father.
- It is **reported** that ...

(7) Light verbs are verbs that carry little semantic content of their own and form a predicate with some additional expression, which is usually a noun. Common verbs that can function as light verbs are: do, give, have, make, take, etc. We group a light verb with subsequent expressions that provide full semantic contents and annotate an entire expression as a single unit.
- She **had** a **smoke**. ("had a smoke" is a light verb construction; we remove "a" from annotation)
- Who will **give** you a **hug**?
- Only the business **made** a **profit**.
- I **got blisters** on my right leg.
- He **won** a **victory** over them.
- Before **making** an important **decision**, he **took** a **shower**. ("took a shower" and "make a decision" are light verb constructions; we remove "important" from the latter because it is a mere specifier for "decision"; see Section A.6 about annotating adjectives)

(8) Phrasal verbs are verbs that made up a main verb together with another element such as an adverb, a preposition, or both. As described at the beginning of this section, we suggest examining contexts and looking up a dictionary in order to determine a verbal phrase.
- She has always **looked down on** me.
- Our party **speaks for** the poor in the country.
- He **spoke of** you with high praise and warm affection.
- I **spoke** with him. (Annotate "spoke" only because "with him" forms a prepositional phrase)
- I **looked over** the writing assignment. (Annotate "looked over" as a single event nugget because it is a phrasal verb)

- I **looked** over my shoulder. (Don't annotate "looked over" but "looked" only; "over my shoulder" is a prepositional phrase)
- I **looked at** him.
- I **looked into** his eyes.
- The class **consists of** 25 students.
- The class **consists in** a hands-on introduction to linear algebra.
- People could be **put in jail** in Denmark for burning the Koran.
- My mother **cried** when my grandpa **kicked** the **bucket**. (Annotate "kicked ... bucket" because "kick the bucket" is an idiomatic verbal phrase meaning "die")
- My mother was **wishing** my grandpa to **die** a good **death**. (Annotate "die ... death" because "die a death" is an idiomatic verbal phrase meaning "die")
- You can **catch up with** the class and should never **give up**.

In addition to the dictionary-lookup strategy described above, we suggest two linguistic tests to decide whether a posed expression is a phrasal verb: (1) the movement test and (2) the conjoining test. The movement test is to move a prepositional phrase into the front of a sentence. If the sentence still makes sense, the expression is a prepositional phrase.

- *Over the writing assignment, I looked. (We cannot say this; "looked over" is a phrasal verb)
- Over my shoulder, I **looked**. (We can say this; "over my shoulder" is a prepositional phrase; "looked over" is not a phrasal verb)

The conjoining test is to conjoin two sentences. If the sentence still makes sense, the two expressions are prepositional phrases.

- *I looked over the writing assignment and my shoulder. (We cannot say this because the first one is a phrasal verb, the second one is a prepositional phrase, and their meanings are different)
- *The class consists of 25 students and in a hands-on introduction to linear algebra. (We cannot say this; "consists of" and "consists in" are phrasal verbs; we cannot separate "consists" and "in" as above)
- I **looked at** him and **into** his eyes. (We can say this; "looked at" and "looked into" are phrasal verbs with similar meanings; annotate "looked at" as a single event nugget and then annotate "looked ... into" as another single (discontinuous) event nugget)

Note that the above sentences "I looked at him" and "I looked into his eyes" fail to pass the movement test, because we cannot say "At him, I looked" or "Into his eyes, I looked." Therefore, they are phrasal verbs. However, unlike the "look over" case, the phrasal verbs "look(ed) at" and "look(ed) into" can pass the conjoining test.

(9) Stative verbs are verbs that express a state rather than an action. Stative verbs include copular verbs, which are verbs that connect an adjective or a noun complement to a subject (e.g., become, get, smell, etc.). One caveat is that we do not annotate copular verbs that indicate mental recognition and have a meaning close to a be-verb (e.g., seem, look, appear). We do not annotate stative verbs that merely refer to attributes of someone or something, playing the almost same role as a be-verb. See the end of this section for these non-eventive cases. Examples of eventive stative verbs are: have, own, hold, need, require, lack, want, love, hate, become, etc. We annotate stative verbs that imply some actions in the past or occurring continuously based on contexts.

- Everybody **liked** her. (Annotate "liked" because it implies the actual occurrence that everybody found her attractive)
- He **lives** in Chicago for most of his life. (Annotate "lives" because it implies numerous things have occurred around him during his residency in Chicago)
- The United States **have** 50 states. (Annotate "have" because it means the possession of the U.S. and implies the action that the U.S. acquired and established 50 states in the past)
- She *has* a good personality. (Don't annotate "has" because it essentially means that she is a good person, which is her attribute)
- He **got sick** yesterday.
- The tomato has **become rotten**.
- She **became** a writer. ("became" is a copular verb but more like an independent main verb that involves a change of her professional status; annotate "became" without grouping it with "writer")
- The milk **turned sour**.
- The stew **smells good**.
- The night **grew dark**.

(10) Occurrence verbs are verbs that express an occurrence of events. Examples are: happen, occur, take place, etc. We annotate an occurrence verb as a single event nugget.

- The **bombing occurred** last Wednesday. (Annotate "occurred" because it is an occurrence verb)
- The **seminar** will **take place** at 4pm. (Annotate "take place" because it is an occurrence verb)

Next, we describe what verbs we do not annotate. Below is a list of types of verbs that we do not annotate:

- auxiliary/modal verbs (e.g., will, can, may, shall, would, could, might, should)
- auxiliary verbs (e.g., have, has, had)
- be-verbs (e.g., am, are, is, was, were, been)
- copular verbs that indicate mental recognition and have a meaning close to a be-verb (e.g., seem, look, appear)
- verbs that play a semantically equivalent role as be-verbs (e.g., mean, equal)

We give several examples of these non-eventive verbs below. For clarification, we also include examples of eventive verbs to compare them with the non-eventive ones.

- John *will be* **coming** soon. ("will" is a modal verb; "be" is a be-verb; annotate "coming" only)
- Mary *has* **married** with Tom. ("has" is an auxiliary verb; annotate "married" only)
- He *might have been* **injured**.
- It *seems* that he **won** the game. ("seems" is a copular verb that indicates mental recognition.)
- She *looks* like a very happy woman.
- He *looked* **tired** after the work. (Don't annotate "looked" because it is a copular verb that indicates mental recognition; annotate "tired" because the adjective implies the action that the work made him tired; see Section A.6 about annotating adjectives for details)
- She *has* a good personality. (Don't annotate "has" because it essentially means that she is a good person, which is her attribute)

- The word caldera *comes* from the Portuguese language, meaning "cauldron." (Don't annotate "comes" and "meaning" because they play the almost same role as a be-verb)
- 'Enormous' *means* 'very big' (Don't annotate "means" because it plays the almost same role as a be-verb, implying no actions)
- Do you know what I **mean**? (Annotate "mean" because it indicates the action "intend to say" and is more like a communication verb)
- Three times two *equals* six. (Don't annotate "equals" because it plays the almost same role as a be-verb, implying no actions)

A.5 Annotating Eventive Nouns

This section provides guidelines about how to annotate eventive nouns and what nouns not to annotate. We annotate nouns when they refer to events. There are some nouns which we should pay special attention to. We explain those nouns in detail as well.

(1) Noun phrases

We consider a noun phrase as a single unit and decide whether the noun phrase refers to an event. As with verb phrases, we suggest examining contexts and looking up a collocation dictionary such as Oxford, WordNet or Wikipedia in order to check the strength of collocation and determine a noun phrase.

- The news **describes** the **shipping accident**. (Annotate the entire phrase "shipping accident" as a single event nugget because it forms a noun phrase referring to an event)
- Community members **provided** information about the general **cleanup process**.
- He has **recovered** from a **heart attack**. (Annotate the entire phrase "heart attack" because the collocation is strong)
- The guerrilla **used** 70 *assault rifles*. ("assault rifles" is a single noun phrase referring to a non-eventive object; "assault" can be the action of attacking someone, but it implies no actions in this context)
- Thousands of *flood victims* were evacuated. ("flood victims" is a single noun phrase referring to people affected by flood; "flood" can be an event, but it implies no actions in this context)
- The *Indian Removal Act* was **signed** into law in 1830. ("Indian Removal Act" is a single noun phrase referring to a law; "removal" can be an event, but it does not imply any actual actions in this context)
- We **faced** massive **oil spill**. ("oil spill" is considered as a single noun phrase because the collocation is strong; "massive" is an adjective which is a mere specifier for "oil spill"; see details for Section A.6 about annotating adjectives)
- We **saw** a **laughing** child. ("laughing" is an adjective modifying "child" and originated from verb "laugh"; it implies the action of laughing; see the section of annotating adjectives)

(2) Proper nouns

We annotate a proper noun as a single event nugget if it refers to an event. Wikipedia is often helpful to determine whether a particular proper noun is an event or not. Some proper nouns appear as metonyms, which are substitutes for other proper nouns. We also annotate such

proper nouns if they refer to an event.

- Property damage by **Hurricane Katrina** was around $108 billion.
- Exactly ten years after **Katrina**, ... (Annotate "Katrina" because in this case it is a metonym referring to "Hurricane Katrina," which is eventive)
- *New Delhi* **announced** today that ... (Don't annotate "New Delhi" because it is a metonym referring to the government of India which is not eventive)

(3) Pronouns or anaphors

We annotate both pronouns and anaphors if they refer to events.

- **It** was **one** of the first **well-known massacres** at school in the United States ... ("It" is a pronoun referring to "Columbine High School Massacre" in a previous context, which is eventive; "one" is the same as "it")
- **This** is ... ("This" is an anaphor, and we annotate it if it corefers to an event)
- The **Boston Tea Party** was **one** of the main **things** that **started** the **American Revolutionary War**.

(4) Empty nouns

When noun phrases consist of empty nouns (e.g., act, action, activity, affair, event, incident, etc.) and taggable modifying nouns, we annotate the entire phrase as one event nugget.

- As an **act of protest** against the Chinese government, ...
- Two **incidents of sexual harassment** have been found.
- This **state of affairs** cannot be **ignored**.

(5) Preposition + noun

If a prepositional phrase indicates a state implying actions, we annotate the entire phrase as an event nugget.

- The people were **in pain**.
- All we **need** is a car, we'll **stay in business**. ("in business" is the original preposition-noun phrase, and "stay in business" is an idiomatic expression)
- The diabetes drugs are currently **under development**.

(6) Nouns that describe states

Some nouns describe states. Some nominal states can imply actions, but others can refer to attributes. For details of the distinction between these two, see the section about annotating adjectives.

- When she **won**, her eyes **shone** with **happiness**. (Annotate "happiness" because it implies the action that her victory made her happy)
- Their grandchildren are a constant source of *happiness*. (Don't annotate "happiness" because it refers to an attribute of some people who have grandchildren)
- Heavy **snow** has **caused chaos** on the highways. (Annotate "chaos" because it implies the confusion of people and cars)

(7) Nouns that are difficult to define as events

Some nouns are ambiguous in terms of eventiveness, and other nouns are simply difficult to decide eventiveness. The former case (ambiguous nouns) include some verb nominalizations (e.g., "payment" in the example below). When we encounter those nouns, we should make a decision considering the context where they are used. If they sound eventive and/or make a semantically meaningful unit with other mentions, we annotate them as event nuggets.

- The report **calls for** a ban on the **import** of illegal drugs. (Annotate "import" because it refers to the action of bringing the drugs)

- Every year lots of *imports* are **brought** from abroad. (Don't annotate "imports" because they refer to objects that brought into a country from abroad)
- I **want** to **report** the **loss** of a package. (Annotate "loss" because it refers to the action of losing the package)
- The *loss* was more than a half of the company's revenue. (Don't annotate "loss" because it refers to the amount of lost money, which is compared to the revenue, rather than the action of losing)
- His **payment** was late. (Annotate "payment" because it refers to the action of his paying something)
- His *payment* was $10. (Don't annotate "payment" because it refers to a specific amount of money paid by him, which is not eventive)
- The report **criticized** the department's **waste** of resources. (Annotate "waste" because it refers to the action of wasting resources)
- We need to **discard** the toxic *wastes*. (Don't annotate "waste" because it refers to some materials that are thrown away)
- **Force** equals mass times acceleration. (This is a difficult case; annotators use their own discretion to examine contexts and decide whether "force" refers to an event)

A.6 Annotating Eventive Adjectives

This section provides guidelines about how to annotate eventive adjectives and what adjectives not to annotate. We follow the definition of eventive adjectives by Palmer et al. (2016). They define eventive adjectives as follows. An attribute (adjective) is an event when its use implies actual occurrences — such as the events leading up to its own existence. Adjectives used as mere specifiers should therefore be viewed with skepticism in this regard. "I came home and saw the door was open" evokes an act of someone opening it; "He walked through the open door" does not. They give the examples below, suggesting a continuous range from eventives to non-eventives:

- The walls **yellowed** during the fire. ("yellowed" is very eventive)
- We **came home** to find the door **opened**. ("opened" is eventive)
- We **came home** to find the door **open**. ("open" is somewhat eventive)
- I **own** a *yellow* canary. ("yellow" is very non-eventive)

Among the examples above, the first two examples (i.e., "yellowed" and "opened") can be recognized as events very easily because they are participles originated from verbs. However, the word "open" in the third example is more difficult to determine. As mentioned above, the "open" is an event because it implies that it occurred after somebody opened the door. In the fourth example, "yellow" is not considered as an event because it is an attribute of the canary and does not evoke any act of making a canary yellow.

The following sentences include eventive adjectives, which should be annotated.

- Tom was **happy** when he **received** a birthday present. ("happy" implies that Tom's reception of the present made Tom happy)
- She was **talkative** at the **party**. ("talkative" implies that she talked a lot at the party)
- He is **blind** to Mary's faults. ("blind" implies that he does not recognize Mary's faults)

- It was quite **unbelievable** that he **won** the game. ("unbelievable" implies the action that people cannot believe that he won the game)
- The bear was **dangerous** and **violent** when he **saw** us. ("dangerous" and "violent" implies that we made the bear upset)
- She was **cradling** a **crying** baby. ("crying" is an adjective implying the action, originated from verb "cry")
- She **made** a **dismissive reply** to his email.
- **It** was the largest **known explosive eruption** within the last 25 million years.

In the examples above, special attention needs to be paid to the last two. In the second to the last, we annotate "made ... dismissive reply" as a single event and do not annotate "made ... reply" and "dismissive" separately. This is because "dismissive" represents a manner of the action "reply," and is originated from verb "dismiss" meaning the action of putting little importance on something, i.e., its own event semantics distinct from the action of "reply." Therefore, when an adjective represents a manner of an eventive noun and implies an independent action (often originated from a verb), we annotate the adjective and noun together as a single event nugget. The annotation of "made ... dismissive reply" is contrastive to "made a quick reply" where we annotate "made ... reply" only, because "quick" is a mere specifier for "reply" and does not imply any occurrences by itself.

Compared with the above examples, the sentences below do not contain any eventive adjectives. Those adjectives only describe attributes of entities rather than implying actions. Therefore, we do not annotate them.

- Tom is a *happy* man. (Don't annotate "happy" because it indicates Tom's attribute)
- Bears can be *dangerous* and *violent* when they see people. ("dangerous" and "violent" express properties of bears with a condition, rather than underlying actions)
- **Stay away** from the volcano because it is *dangerous*. ("dangerous" is not eventive because it depicts the volcano, implying no actual occurrences)
- One of his parents was *blind*. ("blind" is a personal attribute of the parent, implying no actions)
- John **bought** an *expensive* book. (Don't annotate "expensive" because it is a mere specifier for "book" without implying any actions)
- He was the *tallest* in his class.attribute, implying no actions)
- She is a *talkative* person.
- The *rapid* and *massive* **industrialization started** by Stalin ... ("rapid" and "massive" are mere specifiers, implying no actual occurrences)
- They were **waiting** in a *long* line for a **flu shot**.
- ... the **collapse** of land surface after a *gigantic volcanic* **eruption**.

There is an additional note about the construction of "the + adjective." The phrase of "the + adjective" means some people with the state expressed by the adjective. We annotate the phrase when it implies events.

- After the **bombing**, the security guard **found** the **dead** and **injured**. (Annotate both "dead" and "injured" because the dead and injured people imply that they died and got injured due to the bombing)
- We **saw** the *poor* near the station. (Don't annotate "poor" because in this case "poor" is a

mere specifier and does not imply any actual occurrences)

A.7 Annotating Eventive Adverbs

This section provides guidelines about how to annotate eventive adverbs and what adverbs not to annotate. As described in Section A.6, eventive adjectives imply actual occurrences whereas non-eventive adjectives do not, ending up with mere specifiers. We apply the same distinction to adverbs. By definition, an adverb modifies a verb, describing a manner of a verb, that is, how actions are done. We annotate adverbs and verbs together as single events (semantically meaning units), if the adverbs imply actual occurrences.

- She **replied** to his email **dismissively**. (Annotate "replied ... dismissively" as a single event because "dismissively" indicates a manner of the action "replied" and is originated from verb "dismiss" meaning the action of putting little importance on something, i.e., its own event semantics distinct from the action of "replied")
- He **kicked** me **intentionally**.
- He **kicked** me **unintentionally**.
- She **looked at** her **doubtfully**.
- She **looked at** her **undoubtedly**.
- He **closed** the door **angrily** when he was **speaking** with her over the phone. (Annotate "closed ... angrily" is a single event; "angrily" is an adverb implying the action that he got angry due to his phone conversation with her)

On the other hand, we do not annotate adverbs that do not imply any actions and are used as mere specifiers. Such non-eventive adverbs include but are not limited to:

- Adverbs representing times (e.g., early, recently, lately)
- Adverbs representing locations (e.g., somewhere, anywhere)
- Adverbs representing degrees and/or frequencies (e.g., rarely, often, sometimes)
- Adverbs coming at the beginning of a sentence and modify the entire sentence (e.g., luckily, unfortunately, certainly)

Below are examples of non-eventive adverbs:

- Mary *often* **gets up** *early*.
- He **walked** very *slowly*.
- *Actually*, John **did it** on time. ("Actually" modifies the entire sentence without implying any actions; we assume "it" refers to an event)
- *Sadly*, he **opened** the door.
- He *angrily* **closed** the door. (Don't annotate "angrily" if it is unclear whether the adverb implies an actual occurrence from this narrow context)
- He *quietly* **opened** the door.
- The baby **slept** *peacefully*.
- Mary **dressed** *elegantly*.
- Bill **solved** the problem *intelligently*.

Bibliography

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of NAACL-HLT*, pages 19–27.

Aguilar, J., Beller, C., McNamee, P., Durme, B. V., Strassel, S., Song, Z., and Ellis, J. (2014). A comparison of the events and relations across ACE, ERE, TAC-KBP, and FrameNet annotation standards. In *Proceedings of ACL Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 45–53.

Ahn, D. (2006). The stages of event extraction. In *Proceedings of COLING/ACL Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Allan, J. (2002). *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers.

Allan, J., Carbonell, J., Doddington, G., Yamron, J., and Yang, Y. (1998). Topic detection and tracking pilot study final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Allen, J. F. and Ferguson, G. (1994). Actions and events in interval temporal logic. Technical report, University of Rochester.

Anderson, R. C. and Biddle, W. B. (1975). On asking people questions about what they are reading. *Psychology of Learning and Motivation*, 9:90–132.

Andre, T. (1979). Does answering higher level questions while reading facilitate productive learning? *Review of Educational Research*, 49(2):280–318.

Araki, J., Hovy, E., and Mitamura, T. (2014a). Evaluation for partial event coreference. In *Proceedings of ACL Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 68–76.

Araki, J., Liu, Z., Hovy, E., and Mitamura, T. (2014b). Detecting subevent structure for event coreference resolution. In *Proceedings of LREC*, pages 4553–4558.

Araki, J. and Mitamura, T. (2015). Joint event trigger identification and event coreference resolution with structured perceptron. In *Proceedings of EMNLP*, pages 2074–2080.

Araki, J. and Mitamura, T. (2018). Open-domain event detection using distant supervision. In *Proceedings of COLING*.

Araki, J., Rajagopal, D., Sankaranarayanan, S., Holm, S., Yamakawa, Y., and Mitamura, T. (2016). Generating questions and multiple-choice answers using semantic analysis of texts. In *Proceedings of COLING*, pages 1125–1136.

Asher, N. and Lascarides, A. (1998). Bridging. *Journal of Semantics*, 15(1):83–113.

Asher, N. and Lascarides, A. (2003). *Logics of Conversation*. Cambridge University Press.

Auli, M., Galley, M., Quirk, C., and Zweig, G. (2013). Joint language and translation modeling with recurrent neural networks. In *Proceedings of EMNLP*, pages 1044–1054.

Bach, E. (1986). The algebra of events. *Linguistics and Philosophy*, 9:5–16.

Bagga, A. and Baldwin, B. (1998). Algorithms for scoring coreference chains. In *Proceedings of LREC Workshop on Linguistics Coreference*, pages 563–566.

Bagga, A. and Baldwin, B. (1999). Cross-document event coreference: Annotations, experiments, and observations. In *Proceedings of ACL Workshop on Coreference and Its Applications*, pages 1–8.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of COLING*, pages 86–90.

Balasubramanian, N., Soderland, S., Mausam, and Etzioni, O. (2013). Generating coherent event schemas at scale. In *Proceedings of EMNLP*, pages 1721–1731.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2670–2676.

BBN Technologies (2006). *Co-reference Guidelines for English OntoNotes*. BBN Technologies.

Becker, L., Basu, S., and Vanderwende, L. (2012). Mind the gap: Learning to choose gaps for question generation. In *Proceedings of NAACL-HLT*, pages 742–751.

Bejan, C. and Harabagiu, S. (2008). A linguistic resource for discovering event structures and resolving event coreference. In *Proceedings of LREC*.

Bejan, C. and Harabagiu, S. (2010). Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of ACL*, pages 1412–1422.

Bejan, C. and Harabagiu, S. (2014). Unsupervised event coreference resolution. *Computational Linguistics*, 40(2):311–347.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bengtson, E. and Roth, D. (2008). Understanding the value of features for coreference resolution. In *Proceedings of EMNLP*, pages 294–303.

Berant, J., Srikumar, V., Chen, P.-C., Vander Linden, A., Harding, B., Huang, B., Clark, P., and Manning, C. D. (2014). Modeling biological processes for reading comprehension. In *Proceedings of EMNLP*, pages 1499–1510.

Bethard, S., Ogren, P., and Becker, L. (2014). ClearTK 2.0: Design patterns for machine learning in UIMA. In *Proceedings of LREC*, pages 3289–3293.

Bikel, D. M. and Castelli, V. (2008). Event matching using the transitive closure of dependency relations. In *Proceedings of ACL*, pages 145–148.

Bille, P. (2005). A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239.

Bird, S., Loper, E., and Klein, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

Björkelund, A. and Farkas, R. (2012). Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of EMNLP/CoNLL*, pages 49–55.

Björkelund, A., Hafdell, L., and Nugues, P. (2009). Multilingual semantic role labeling. In *Proceedings of CoNLL*, pages 43–48.

Björkelund, A. and Kuhn, J. (2014). Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of ACL*, pages 47–57.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*, pages 92–100.

Bohnet, B. and Nivre, J. (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP/CoNLL*, pages 1455–1465.

Bronstein, O., Dagan, I., Li, Q., Ji, H., and Frank, A. (2015). Seed-based event trigger labeling: How far can event descriptions get us? In *Proceedings of ACL/IJCNLP*, pages 372–376.

Cai, J. and Strube, M. (2010). Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of SIGDIAL 2010*, pages 28–36.

Campbell, N. and Reece, J. (2005). *Biology*. Benjamin Cummings.

Cao, K., Li, X., Fan, M., and Grishman, R. (2015). Improving event detection with active learning. In *Proceedings of RANLP*, pages 72–77.

Carlson, L. (1981). Aspect and quantification. *Syntax and Semantics*, 14:31–64.

Chambers, N., Cassidy, T., McDowell, B., and Bethard, S. (2014). Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Chambers, N. and Jurafsky, D. (2008). Unsupervised learning of narrative event chains. In *Proceedings of ACL-HLT*, pages 789–797.

Chambers, N. and Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL/IJCNLP*, pages 602–610.

Chen, W., Aist, G., and Mostow, J. (2009a). Generating questions automatically from informational text. In *Proceedings of the 2nd Question Generation Workshop*.

Chen, Y., Xu, L., Liu, K., Zeng, D., and Zhao, J. (2015). Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL/IJCNLP*, pages 167–176.

Chen, Z., Ji, H., and Haralick, R. (2009b). A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of RANLP Workshop on Events in Emerging Text Types*, pages 17–22.

Chiu, J. and Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Chklovski, T. and Pantel, P. (2004). VerbOcean: Mining the Web for fine-grained semantic verb relations. In *Proceedings of EMNLP*, pages 33–40.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Choubey, P. K. and Huang, R. (2017). Event coreference resolution by iteratively unfolding inter-dependencies among events. In *Proceedings of EMNLP*.

Chung, S. and Timberlake, A. (1985). Tense, mood and aspect. In *Language Typology and Syntactic Description: Volume 3, Grammatical Categories and the Lexicon*. Cambridge University Press.

Clark, H. H. (1977). Bridging. In Johnson-Laird, P. N. and Wason, P. C., editors, *Thinking: Readings in Cognitive Science*. Cambridge.

Clark, S., Curran, J., and Osborne, M. (2003). Bootstrapping POS-taggers using unlabelled data. In *Proceedings of CoNLL*, pages 49–55.

Collins, M. (2002). Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.

Collins, M. and Duffy, N. (2001). Convolution kernels for natural language. In *Proceedings of NIPS*, pages 625–632.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL*, pages 263–270.

Collins, M. and Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 111–118.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Craven, M. and Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pages 77–86.

Croce, D., Moschitti, A., and Basili, R. (2011). Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of EMNLP*, pages 1034–1046.

Curto, S., Mendes, A. C., and Coheur, L. (2011). Exploring linguistically-rich patterns for question generation. In *Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop*, pages 33–38.

Cybulska, A. and Vossen, P. (2012). Using semantic relations to solve event coreference in text. In *Proceedings of LREC Workshop on Semantic Relations-II Enhancing Resources and Applications*, pages 60–67.

Cybulska, A. and Vossen, P. (2014). Guidelines for ECB+ annotation of events and their coref-

erence. Technical Report NWR-2014-1, VU University Amsterdam.

Danlos, L. (2001). Event coreference between two sentences. In Bunt, H., Muskens, R., and Thijsse, E., editors, *Computing Meaning*, volume 2, pages 271–288.

DARPA (2012). Deep exploration and filtering of text (deft).

Das, D., Chen, D., Martins, A. F. T., Schneider, N., and Smith, N. A. (2014). Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010). Probabilistic frame-semantic parsing. In *Proceedings of NAACL-HLT*, pages 948–956.

Daumé III, H. (2008). Cross-task knowledge-constrained self training. In *Proceedings of EMNLP*, pages 680–688.

de Marneffe, M.-C., Rafferty, A. N., and Manning, C. D. (2008). Finding contradictions in text. In *Proceedings of ACL-HLT*, pages 1039–1047.

Demaine, E. D., Mozes, S., Rossman, B., and Weimann, O. (2009). An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms*, 6(1):2:1–2:19.

Denis, P. and Baldridge, J. (2009). Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42:87–96.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of ACL*, pages 1370–1380.

Ding, X., Zhang, Y., Liu, T., and Duan, J. (2014). Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of EMNLP*, pages 1415–1425.

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., and Weischedel, R. (2004). The automatic content extraction (ACE) program tasks, data, and evaluation. In *Proceedings of LREC*, pages 837–840.

Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of SIGKDD*, pages 155–164.

dos Santos, C., Xiang, B., and Zhou, B. (2015). Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL/IJCNLP*, pages 626–634.

dos Santos, C. N. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING*, pages 69–78.

dos Santos, C. N. and Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of ICML*, pages 1818–1826.

Durrett, G., Hall, D., and Klein, D. (2013). Decentralized entity-level modeling for coreference resolution. In *Proceedings of ACL*, pages 114–124.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Exner, P., Klang, M., and Nugues, P. (2015). A distant supervision approach to semantic role labeling. In *Proceedings of *SEM*, pages 239–248.

Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information

extraction. In *Proceedings of EMNLP*, pages 1535–1545.

Feng, X., Huang, L., Tang, D., Ji, H., Qin, B., and Liu, T. (2016). A language-independent neural network for event detection. In *Proceedings of ACL*, pages 66–71.

Fernandes, E., dos Santos, C. N., and Milidiú, R. L. (2012). Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of EMNLP/CoNLL*, pages 41–48.

Filatova, E. and Hatzivassiloglou, V. (2003). Domain-independent detection, extraction, and labeling of atomic events. In *Proceedings of RANLP*, pages 145–152.

Filatova, E. and Hatzivassiloglou, V. (2004). Event-based extractive summarization. In *Proceedings of ACL Workshop: Text Summarization Branches Out*, pages 104–111.

Fillmore, C. J. (1976). Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, 280(1):20–32.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

Florian, R., Pitrelli, J. F., Roukos, S., and Zitouni, I. (2010). Improving mention detection robustness to noisy input. In *Proceedings of EMNLP*, pages 335–345.

Frermann, L., Titov, I., and Pinkal, M. (2014). A hierarchical Bayesian model for unsupervised induction of script knowledge. In *Proceedings of EACL*, pages 49–57.

Ghaeini, R., Fern, X., Huang, L., and Tadepalli, P. (2016). Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of ACL*, pages 369–373.

Glavaš, G. and Šnajder, J. (2013a). Exploring coreference uncertainty of generically extracted event mentions. In *Proceedings of CICLing*, pages 408–422.

Glavaš, G. and Šnajder, J. (2013b). Recognizing identical events with graph kernels. In *Proceedings of ACL*, pages 797–803.

Goyal, K., Jauhar, S. K., Li, H., Sachan, M., Srivastava, S., and Hovy, E. (2013). A structured distributional semantic model for event co-reference. In *Proceedings of ACL*, pages 467–473.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Grishman, R. and Sundheim, B. (1996). Message understanding conference-6: A brief history. In *Proceedings of COLING*, pages 466–471.

Grishman, R., Westbrook, D., and Meyers, A. (2005). NYU's English ACE 2005 system description. In *Proceedings of ACE 2005 Evaluation Workshop*.

Gung, J. and Kalita, J. (2012). Summarization of historical articles using temporal event clustering. In *Proceedings of NAACL-HLT*, pages 631–635.

Haghighi, A. D., Ng, A. Y., and Manning, C. D. (2005). Robust textual inference via graph matching. In *Proceedings of HLT/EMNLP*, pages 387–394.

Hamaker, C. (1986). The effect of adjunct questions on prose learning. *Review of Educational Research*, 56(2):212–242.

Hardy, H., Kanchakouskaya, V., and Strzalkowski, T. (2006). Automatic event classification using surface text features. In *Proceedings of AAAI Workshop on Event Extraction and Synthesis*, pages 36–41.

Haviland, S. E. and Clark, H. H. (1974). What's new? Acquiring new information as a process in comprehension. *Journal of Verbal Learning and Verbal Behavior*, 13:512–521.

Heilman, M. and Smith, N. A. (2010a). Good question! Statistical ranking for question generation. In *Proceedings of NAACL-HLT*, pages 609–617.

Heilman, M. and Smith, N. A. (2010b). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL-HLT*, pages 1011–1019.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Proceedings of NIPS*, pages 1693–1701.

Hirst, G. and St-Onge, D. (1998). Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. In Fellbaum, C., editor, *WordNet: An Electronic Lexical Database*, pages 305–332.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hong, Y., Lu, D., Yu, D., Pan, X., Wang, X., Chen, Y., Huang, L., and Ji, H. (2015). RPI_BLENDER TAC-KBP2015 system description. In *Proceedings of Text Analysis Conference 2015*.

Hong, Y., Zhang, J., Ma, B., Yao, J., Zhou, G., and Zhu, Q. (2011). Using cross-entity inference to improve event extraction. In *Proceedings of ACL-HLT*, pages 1127–1136.

Hovy, D., Plank, B., Alonso, H. M., and Søgaard, A. (2015). Mining for unambiguous instances to adapt part-of-speech taggers to new domains. In *Proceedings of NAACL-HLT*, pages 1256–1261.

Hovy, E., Mitamura, T., Verdejo, F., Araki, J., and Philpot, A. (2013). Events are not simple: Identity, non-identity, and quasi-identity. In *Proceedings of NAACL-HLT Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 21–28.

Huang, L., Fayong, S., and Guo, Y. (2012). Structured perceptron with inexact search. In *Proceedings of NAACL-HLT*, pages 142–151.

Huang, R. and Riloff, E. (2012). Bootstrapped training of event extraction classifiers. In *Proceedings of EACL*, pages 286–295.

Huang, Z., Eidelman, V., and Harper, M. (2009). Improving a simple bigram HMM part-of-speech tagger by latent annotation and self-training. In *Proceedings of NAACL-HLT*, pages 213–216.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1502.06922*.

Humphreys, K., Gaizauskas, R., and Azzam, S. (1997). Event coreference for information extraction. In *Proceedings of ACL/EACL Workshop on Operational Factors in Practical, Robust*

Anaphora Resolution for Unrestricted Texts, pages 75–81.

Huttunen, S., Yangarber, R., and Grishman, R. (2002). Complexity of event structure in IE scenarios. In *Proceedings of COLING*, pages 1–7.

Irmer, M. (2008). Bridges between events. Indirect reference to eventualities. In Benz, A., Kühnlein, P., and Stede, M., editors, *Proceedings of Constraints in Discourse III (CID-08)*, pages 103–110.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and III, H. D. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL/IJCNLP*, pages 1681–1691.

Jans, B., Bethard, S., Vulić, I., and Moens, M.-F. (2012). Skip n-grams and ranking functions for predicting script events. In *Proceedings of EACL*, pages 336–344.

Ji, H. (2009). Cross-lingual predicate cluster acquisition to improve bilingual event extraction by inductive learning. In *Proceedings of NAACL-HLT Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*, pages 27–35.

Ji, H. and Grishman, R. (2008). Refining event extraction through cross-document inference. In *Proceedings of ACL-HLT*, pages 254–262.

Ji, H. and Grishman, R. (2011). Knowledge base population: Successful approaches and challenges. In *Proceedings of ACL-HLT*, pages 1148–1158.

Ji, H., Westbrook, D., and Grishman, R. (2005). Using semantic relations to refine coreference decisions. In *Proceedings of HLT/EMNLP*, pages 17–24.

Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING X*, pages 19–33.

Johansson, R. and Nugues, P. (2008). Dependency-based semantic role labeling of PropBank. In *Proceedings of EMNLP*, pages 69–78.

Kim, J., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. (2009). Overview of BioNLP'09 shared task on event extraction. In *Proceedings of BioNLP-ST Workshop*, pages 1–9.

Kim, J., Wang, Y., and Yamamoto, Y. (2013). The Genia Event Extraction shared task, 2013 edition - overview. In *Proceedings of BioNLP-ST Workshop*, pages 8–15.

Kim, J.-D., Ohta, T., and Tsujii, J. (2008). Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9:10.

Kim, J.-D., Wang, Y., Takagi, T., and Yonezawa, A. (2011). Overview of genia event task in BioNLP shared task 2011. In *Proceedings of BioNLP-ST Workshop*, pages 7–15.

Kimmig, A., Bach, S. H., Broecheler, M., Huang, B., and Getoor, L. (2012). A short introduction to probabilistic soft logic. In *Proceedings of NIPS Workshop on Probabilistic Programming: Foundations and Applications*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Kipper, K., Korhonen, A., Ryant, N., and Palmer, M. (2008). A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Klein, P. N. (1998). Computing the edit-distance between unrooted ordered trees. In *Proceedings of ESA 1998*, pages 91–102.

Krause, S., Xu, F., Uszkoreit, H., and Weissenborn, D. (2016). Event linking with sentential features from convolutional neural networks. In *Proceedings of CoNLL*, pages 239–249.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Lassalle, E. and Denis, P. (2013). Improving pairwise coreference models through feature space hierarchy learning. In *Proceedings of ACL*, pages 497–506.

LDC (2005). *ACE (Automatic Content Extraction) English Annotation Guidelines for Events*. Linguistic Data Consortium.

LDC (2015). *DEFT Rich ERE Annotation Guidelines: Events*. Linguistic Data Consortium.

Leacock, C. and Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. In Fellfaum, C., editor, *WordNet: An Electronic Lexical Database*, pages 265–283.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of IEEE 1998*, 86(11):2278–2324.

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., and Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M., and Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of CoNLL*, pages 28–34.

Lee, H., Recasens, M., Chang, A., Surdeanu, M., and Jurafsky, D. (2012). Joint entity and event coreference resolution across documents. In *Proceedings of EMNLP/CoNLL*, pages 489–500.

Lee, H., Surdeanu, M., Maccartney, B., and Jurafsky, D. (2014). On the importance of text analysis for stock price prediction. In *Proceedings of LREC*, pages 1170–1175.

Lee, K., He, L., Lewis, M., and Zettlemoyer, L. (2017). End-to-end neural coreference resolution. In *Proceedings of EMNLP*, pages 188–197.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC 1986*, pages 24–26.

Levin, B. (1993). *English Verb Classes and Alternation: A Preliminary Investigation*. The University of Chicago Press.

Lewis, M., Lee, K., and Zettlemoyer, L. (2016). LSTM CCG parsing. In *Proceedings of NAACL-HLT*, pages 221–231.

Li, Q. and Ji, H. (2014). Incremental joint extraction of entity mentions and relations. In *Proceedings of ACL*, pages 402–412.

Li, Q., Ji, H., Hong, Y., and Li, S. (2014). Constructing information networks using one single model. In *Proceedings of EMNLP*, pages 1846–1851.

Li, Q., Ji, H., and Huang, L. (2013). Joint event extraction via structured prediction with global features. In *Proceedings of ACL*, pages 73–82.

Li, W., Wu, M., Lu, Q., Xu, W., and Yuan, C. (2006). Extractive summarization using inter- and intra- event relevance. In *Proceedings of COLING/ACL*, pages 369–376.

Liao, S. and Grishman, R. (2010). Using document level cross-event inference to improve event extraction. In *Proceedings of ACL*, pages 789–797.

Liao, S. and Grishman, R. (2011). Can document selection help semi-supervised learning? a case study on event extraction. In *Proceedings of ACL-HLT*, pages 260–265.

Lin, D. (1998). An information-theoretic definition of similarity. In *Proceedings of ICML*, pages 296–304.

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. In *Proceedings of ICLR*.

Lindberg, D., Popowich, F., Nesbit, J., and Winne, P. (2013). Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528.

Liu, M., Li, W., Wu, M., and Lu, Q. (2007). Extractive summarization based on event term clustering. In *Proceedings of ACL*, pages 185–188.

Liu, P., Qiu, X., Chen, J., and Huang, X. (2016a). Deep fusion LSTMs for text semantic matching. In *Proceedings of ACL*, pages 1034–1043.

Liu, S., Chen, Y., He, S., Liu, K., and Zhao, J. (2016b). Leveraging FrameNet to improve automatic event detection. In *Proceedings of ACL*, pages 2134–2143.

Liu, S., Liu, K., He, S., and Zhao, J. (2016c). A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of AAAI*, pages 2993–2999.

Liu, T. and Strzalkowski, T. (2012). Bootstrapping events and relations from text. In *Proceedings of EACL*, pages 296–305.

Liu, Z., Araki, J., Dua, D., Mitamura, T., and Hovy, E. (2015). CMU-LTI at KBP 2015 event track. In *Proceedings of Text Analysis Conference 2015*.

Liu, Z., Araki, J., Hovy, E., and Mitamura, T. (2014). Supervised within-document event coreference using information propagation. In *Proceedings of LREC*, pages 4539–4544.

Lu, J. and Ng, V. (2016). Event coreference resolution with multi-pass sieves. In *Proceedings of LREC*, pages 3996–4003.

Lu, W. and Roth, D. (2012). Automatic event extraction with structured preference modeling. In *Proceedings of ACL*, pages 835–844.

Luo, B., Yang, H., Zeng, Y., Feng, Y., and Zhao, D. (2015). WIP event detection system at TAC KBP 2015 event nugget track. In *Proceedings of Text Analysis Conference 2015*.

Luo, X. (2005). On coreference resolution performance metrics. In *Proceedings of HLT/EMNLP*,

pages 25–32.

Luo, X., Pradhan, S., Recasens, M., and Hovy, E. (2014). An extension of BLANC to system mentions. In *Proceedings of ACL*, pages 24–29.

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of ACL*, pages 1064–1074.

Macleod, C., Grishman, R., Meyers, A., Barrett, L., and Reeves, R. (1998). Nomlex: A lexicon of nominalizations. In *Proceedings of EURALEX 1998*, pages 187–193.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings ACL: System Demonstrations*, pages 55–60.

Manshadi, M., Swanson, R., and Gordon, A. S. (2008). Learning a probabilistic model of event sequences from Internet Weblog stories. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*, pages 159–164.

Marsh, E. and Perzanowski, D. (1998). MUC-7 evaluation of IE technology: Overview of results. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.

Marujo, L., Ling, W., Ribeiro, R., Gershman, A., Carbonell, J., de Matos, D. M., and ao P. Neto, J. (2016). Exploring events and distributed representations of text in multi-document summarization. *Knowledge-Based Systems*, 94:33–42.

Mazidi, K. and Nielsen, R. D. (2014). Linguistic considerations in automatic question generation. In *Proceedings of ACL*, pages 321–326.

McClosky, D., Charniak, E., and Johnson, M. (2006). Effective self-training for parsing. In *Proceedings of NAACL-HLT*, pages 152–159.

McClosky, D., Surdeanu, M., and Manning, C. (2011). Event extraction as dependency parsing. In *Proceedings of ACL-HLT*, pages 1626–1635.

McDonald, R., Pereira, F., Kulick, S., Winters, S., Jin, Y., and White, P. (2005). Simple algorithms for complex relation extraction with applications to biomedical IE. In *Proceedings ACL 2005*, pages 491–498.

McNamee, P. and Dang, H. (2009). Overview of the tac 2009 knowledge base population track. In *Proceedings of TAC Workshop*.

Mehdad, Y. (2009). Automatic cost estimation for tree edit distance using particle swarm optimization. In *Proceedings of ACL/IJCNLP*, pages 289–292.

Mehdad, Y., Moschitti, A., and Zanzotto, F. M. (2010). Syntactic/semantic structures for textual entailment recognition. In *Proceedings of NAACL-HLT*, pages 1020–1028.

Mendes, P., Jakob, M., and Bizer, C. (2012). DBpedia: A multilingual cross-domain knowledge base. In *Proceedings of LREC*.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004). The NomBank project: An interim report. In *Proceedings of HLT-NAACL Workshop: Frontiers in Corpus Annotation*, pages 24–31.

Mikolov, T., Yih, W., and Zweig, G. (2013). Linguistic regularities in continuous space word

representations. In *Proceedings of NAACL-HLT*, pages 746–751.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1990). Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

Miller, G. A., Leacock, C., Tengi, R., and Bunker, R. T. (1993). A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of ACL/IJCNLP*, pages 1003–1011.

Mitamura, T., Liu, Z., and Hovy, E. (2015a). Overview of TAC-KBP 2015 Event Nugget track. In *Proceedings of Text Analysis Conference 2015*.

Mitamura, T., Liu, Z., and Hovy, E. (2016). Overview of TAC-KBP 2016 Event Nugget track. In *Proceedings of Text Analysis Conference*.

Mitamura, T., Liu, Z., and Hovy, E. (2017). Events detection, coreference and sequencing: What's next? Overview of the TAC KBP 2017 Event track. In *Proceedings of Text Analysis Conference*.

Mitamura, T., Yamakawa, Y., Holm, S., Song, Z., Bies, A., Kulick, S., and Strassel, S. (2015b). Event nugget annotation: Processes and issues. In *Proceedings of NAACL-HLT Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 66–76.

Mitkov, R. and Ha, L. (2003). Computer-aided generation of multiple-choice tests. In *Proceedings of NAACL-HLT Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22.

Miwa, M. and Bansal, M. (2016). End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of ACL*, pages 1105–1116.

Modi, A. (2016). Event embeddings for semantic script modeling. In *Proceedings of CoNLL*, pages 75–57.

Modi, A. and Titov, I. (2014). Inducing neural models of script knowledge. In *Proceedings of CoNLL*, pages 49–57.

Moens, M. and Steedman, M. (1988). Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–28.

Monahan, S., Mohler, M., Tomlinson, M., Book, A., Gorelkin, M., Crosby, K., and Brunson, M. (2015). Populating a knowledge base with information about events. In *Proceedings of Text Analysis Conference 2015*.

Moschitti, A., Pighin, D., and Basili, R. (2008). Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Mou, L., Men, R., Li, G., Xu, Y., Zhang, L., Yan, R., and Jin, Z. (2016). Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of ACL*, pages 130–136.

Mourelatos, A. P. (1978). Events, processes, and states. *Linguistics and Philosophy*, 2:415–434.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Ng, V. and Cardie, C. (2002). Improving machine learning approaches to coreference resolution. In *Proceedings of ACL*, pages 104–111.

Nguyen, T. H., Cho, K., and Grishman, R. (2016). Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 300–309.

Nguyen, T. H. and Grishman, R. (2015). Event detection and domain adaptation with convolutional neural networks. In *Proceedings of ACL/IJCNLP*, pages 365–371.

O'Gorman, T., Wright-Bettner, K., and Palmer, M. (2016). Richer Event Description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines*, pages 47–56.

Okazaki, N. (2007). CRFsuite: A fast implementation of conditional random fields (CRFs).

Onyshkevych, B., Okurowski, M. E., and Carlson, L. (1993). Tasks, domains, and languages for information extraction. In *Proceedings of the TIPSTER Text Program: Phase I*, pages 123–133.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Palmer, M., Styler, W., Crooks, K., and O'Gorman, T. (2016). *Richer Event Description (RED) Annotation Guidelines*. University of Colorado at Boulder.

Patwardhan, S. and Riloff, E. (2009). A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of EMNLP*, pages 151–160.

Pawlik, M. and Augsten, N. (2011). RTED: A robust algorithm for the tree edit distance. *Proceedings of PVLDB*, 5(4):334–345.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Pichotta, K. and Mooney, R. (2014). Statistical script learning with multi-argument events. In *Proceedings of EACL*, pages 220–229.

Pichotta, K. and Mooney, R. J. (2016). Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of AAAI*.

Piwek, P. and Krahmer, E. (2000). Presuppositions in context: Constructing bridges. In Bonzon, P., Cavalcanti, M., and Nossum, R., editors, *Formal Aspects of Context*, pages 85–106. Kluwer Academic Publishers.

Poesio, M., Mehta, R., Maroudas, A., and Hitzeman, J. (2004). Learning to resolve bridging references. In *Proceedings of ACL*, pages 143–150.

Poon, H. and Vanderwende, L. (2010). Joint inference for knowledge extraction from biomedical literature. In *Proceedings of NAACL-HLT*, pages 813–821.

Pradhan, S., Luo, X., Recasens, M., Hovy, E., Ng, V., and Strube, M. (2014). Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of ACL*, pages 30–35.

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R., and Xue, N. (2011). CoNLL-2011 shared task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of CoNLL*,

pages 1–27.

Pradhan, S. S., Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2007a). OntoNotes: A unified relational semantic representation. In *Proceedings of the International Conference on Semantic Computing*, pages 517–526.

Pradhan, S. S., Ramshaw, L., Weischedel, R., MacBride, J., and Micciulla, L. (2007b). Unrestricted coreference: Identifying entities and events in OntoNotes. In *Proceedings of the 2007 International Conference on Semantic Computing*, pages 446–453.

Pulman, S. G. (1997). Aspectual shift as type coercion. *Transactions of the Philological Society*, 95(2):279–317.

Pustejovsky, J. (2000). Events and the semantics of opposition. In Tenny, C. and Pustejovsky, J., editors, *Events as Grammatical Objects*. Cambridge University Press.

Pustejovsky, J., Castaño, J. M., Ingria, R., Sauri, R., Gaizauskas, R. J., Setzer, A., and Katz, G. (2003). TimeML: Robust specification of event and temporal expressions in text. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 28–34.

Rahman, A. and Ng, V. (2009). Supervised models for coreference resolution. In *Proceedings of EMNLP*, pages 968–977.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL*, pages 147–155.

Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of ACL*, pages 1375–1384.

Recasens, M. and Hovy, E. (2011). BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Regneri, M., Koller, A., and Pinkal, M. (2010). Learning script knowledge with Web experiments. In *Proceedings of ACL*, pages 979–988.

Regneri, M. and Wang, R. (2012). Using discourse information for paraphrase extraction. In *Proceedings of EMNLP*, pages 916–927.

Reimers, N. and Gurevych, I. (2015). Event nugget detection, classification and coreference resolution using deep neural networks and gradient boosted decision trees. In *Proceedings of Text Analysis Conference 2015*.

Reimers, N. and Gurevych, I. (2017). Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Reschke, K., Jankowiak, M., Surdeanu, M., Manning, C., and Jurafsky, D. (2014). Event extraction using distant supervision. In *Proceedings of LREC*, pages 4527–4531.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI 1995*, pages 448–453.

Riedel, S. and McCallum, A. (2011). Fast and robust joint models for biomedical event extraction. In *Proceedings of EMNLP*, pages 1–12.

Ritter, A., Clark, S., Mausam, and Etzioni, O. (2011). Named entity recognition in tweets: An experimental study. In *Proceedings of EMNLP*, pages 1524–1534.

Ritter, A., Mausam, Etzioni, O., and Clark, S. (2012). Open domain event extraction from Twitter. In *Proceedings of SIGKDD*, pages 1104–1112.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kociský, T., and Blunsom, P. (2016). Reasoning about entailment with neural attention. In *Proceedings of ICLR*.

Roth, M. and Lapata, M. (2016). Neural semantic role labeling with dependency path embeddings. In *Proceedings of ACL*, pages 1192–1202.

Ruder, S., Ghaffari, P., and Breslin, J. G. (2016). A hierarchical model of reviews for aspect-based sentiment analysis. In *Proceedings of EMNLP*, pages 999–1005.

Sachan, M., Hovy, E., and Xing, E. P. (2015). An active learning approach to coreference resolution. In *Proceedings of IJCAI*, pages 1312–1318.

Sagae, K. and Tsujii, J. (2007). Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of EMNLP/CoNLL*, pages 1044–1050.

Sarawagi, S. and Cohen, W. W. (2004). Semi-markov conditional random fields for information extraction. In *Proceedings of NIPS*, pages 1185–1192.

Sarkar, A. (2001). Applying co-training methods to statistical parsing. In *Proceedings of NAACL-HLT*.

Schank, R. C. and Abelson, R. P. (1977). *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum Associates.

Schneider, N., Danchik, E., Dyer, C., and Smith, N. A. (2014). Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Scudder, H. J. (1965). Probability of error of some adaptive pattern-recognition machines. *IEEE Transaction on Information Theory*, 11(3):363–371.

Setzer, A. and Gaizauskas, R. (2000). Annotating events and temporal information in newswire texts. In *Proceedings of LREC*, pages 1287–1294.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of NAACL-HLT*, pages 213–220.

Sha, L., Liu, J., Lin, C.-Y., Li, S., Chang, B., and Sui, Z. (2016). RBPB: Regularization-based pattern balancing method for event extraction. In *Proceedings of ACL*, pages 1224–1234.

Søgaard, A. (2010). Simple semi-supervised training of part-of-speech taggers. In *Proceedings of ACL*, pages 205–208.

Søgaard, A. and Rishøj, C. (2010). Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of COLING*, pages 1065–1073.

Song, Z., Bies, A., Strassel, S., Riese, T., Mott, J., Ellis, J., Wright, J., Kulick, S., Ryant, N., and Ma, X. (2015). From Light to Rich ERE: Annotation of entities, relations, and events. In *Proceedings of NAACL-HLT Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 89–98.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning*

Research, 15:1929–1958.

Srivastava, S., Hovy, D., and Hovy, E. (2013). A walk-based semantically enriched tree kernel over distributed word representations. In *Proceedings of EMNLP*, pages 1411–1416.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: A Web-based tool for NLP-assisted text annotation. In *Proceedings of EACL: Demonstrations Session*, pages 102–107.

Stoyanov, V., Gilbert, N., Cardie, C., and Riloff, E. (2009). Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of ACL/IJCNLP*, pages 656–664.

Sundheim, B. M. (1995). Overview of results of the MUC-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding*, pages 13–31.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Tai, K.-C. (1979). The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433.

Tomadaki, E. and Salway, A. (2005). Matching verb attributes for cross-document event coreference. In *Proceedings of the Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*, pages 127–132.

Tratz, S. and Hovy, E. (2011). A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of EMNLP*, pages 1257–1268.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Vendler, Z. (1957). Verbs and times. *Philosophical Review*, 66(2):143–160.

Venugopal, D., Chen, C., Gogate, V., and Ng, V. (2014). Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of EMNLP*, pages 831–843.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D., and Hirschman, L. (1995). A model-theoretic coreference scoring scheme. In *Proceedings of MUC-6*, pages 45–52.

Walker, C., Strassel, S., Medero, J., and Maeda, K. (2006). ACE 2005 multilingual training corpus.

Wang, M. and Manning, C. D. (2010). Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of COLING*, pages 1164–1172.

Wang, Q. I., Schuurmans, D., and Lin, D. (2008). Semi-supervised convex training for dependency parsing. In *Proceedings of ACL-HLT*, pages 532–540.

Wang, W. and Chang, B. (2016). Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of ACL*, pages 2306–2315.

Wattarujeekrit, T., Shah, P. K., and Collier, N. (2004). PASBio: predicate-argument structures for event extraction in molecular biology. *BMC Bioinformatics*, 5:155.

Weischedel, R., Hovy, E., Marcus, M., Palmer, M., Belvin, R., Pradhan, S., Ramshaw, L., and

Xue, N. (2011). OntoNotes: A large training corpus for enhanced processing. In *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*, pages 54–63. Springer-Verlag New York.

Weiss, D., Alberti, C., Collins, M., and Petrov, S. (2015). Structured training for neural network transition-based parsing. In *Proceedings of ACL/IJCNLP*, pages 323–333.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of ACL*, pages 133–138.

Xu, W., Grishman, R., and Zhao, L. (2011). Passage retrieval for information extraction using distant supervision. In *Proceedings of IJCNLP*, pages 1046–1054.

Yang, B., Cardie, C., and Frazier, P. (2015). A hierarchical distance-dependent Bayesian model for event coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:517–528.

Yang, B. and Mitchell, T. (2016). Joint extraction of events and entities within a document context. In *Proceedings of NAACL-HLT*, pages 289–299.

Yang, W. (1991). Identifying syntactic differences between two programs. *Software: Practice and Experience*, 21(7):739–755.

Yao, K., Zweig, G., Hwang, M.-Y., Shi, Y., and Yu, D. (2013a). Recurrent neural networks for language understanding. In *Proceedings of INTERSPEECH 2013*, pages 2524–2528.

Yao, X., Durme, B. V., Callison-burch, C., and Clark, P. (2013b). Answer extraction as sequence tagging with tree edit distance. In *Proceedings of NAACL-HLT*, pages 858–867.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*, pages 189–196.

Yin, W. and Schütze, H. (2015). Discriminative phrase embedding for paraphrase identification. In *Proceedings of NAACL-HLT*, pages 1368–1373.

Zanzotto, F. M. and Moschitti, A. (2006). Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of COLING/ACL*, pages 401–408.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

Zhang, R., Li, W., and Lu, Q. (2010). Sentence ordering with event-enriched semantics and two-layered clustering for multi-document news summarization. In *Proceedings of COLING*, pages 1489–1497.

Zhang, Y. and Clark, S. (2008a). Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-HLT*, pages 888–896.

Zhang, Y. and Clark, S. (2008b). A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP*, pages 562–571.

Zhong, Z. and Ng, H. T. (2010). It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of ACL: System Demonstrations*, pages 78–83.

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of ACL*,

pages 207–212.

Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.