KNOWLEDGE-RICH EVENT COREFERENCE RESOLUTION

by

Jing Lu

APPROVED BY SUPERVISORY COMMITTEE:

Vincent Ng, Chair

Vibhav Gogate

Jessica Ouyang

Nicholas Ruozzi

KNOWLEDGE-RICH EVENT COREFERENCE RESOLUTION

by

JING LU, BE, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2021

ACKNOWLEDGMENTS

First and foremost, I would like to thank Dr. Vincent Ng for all his guidance and support throughout my PhD journey. I am extremely grateful for all the opportunities given to me to work on different projects and collaborate with different research teams. I am also grateful for all his patience and insightful suggestions when I got stuck on my projects. I will always remember the days and nights spent on discussing research ideas, solving problems and editing papers.

I would like to express my appreciation to Dr. Vibhav Gogate, Dr. Jessica Ouyang and Dr. Nicholas Ruozzi for serving on my committee and providing constructive suggestions on this dissertation.

In addition, I am thankful for having tons of support from my labmates. I would like to thank Chen Chen and Luis Gerardo Mojica for their advice in the early stage of my PhD journey. I would also like to thank Gerardo Ocampo Diaz, Hideo Kobayashi, Yang Yu, Mingqing Ye, Zixuan Ke, Hui Lin, and Yize Pang for all the support and encouragement. Besides working together in ECSS3.414, I enjoyed the birthday parties, lab activities and exploring restaurants around the campus with them.

I would also like to acknowledge my extraordinary mentors from my internship at Google Research for their relentless support. I had great pleasure of working with Andrew Nystrom, Tao Chen, Burcu Karagol Ayan, Gustavo Hernández Ábrego, Ji Ma, Jianmo Ni, and Yinfei Yang.

Finally, I appreciate all the love and encouragement from my family. Especially, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me.

November 2020

KNOWLEDGE-RICH EVENT COREFERENCE RESOLUTION

Jing Lu, PhD
The University of Texas at Dallas, 2021

Supervising Professor: Vincent Ng, Chair

Information extraction, a key area of research in Natural Language Processing (NLP), concerns the extraction of structured information from natural language documents. Recent years have seen a gradual shift of focus from entity-based tasks to event-based tasks in information extraction research. Being a core event-based task, event coreference resolution, the task of determining which event mentions in a document refer to the same real-world event, is generally considered one of the most challenging tasks in NLP. More specifically, for two event mentions to be coreferent, both their triggers (i.e., the words realizing the occurrence of events) and their corresponding arguments (e.g., time, places, and people involved in them) have to be compatible. However, identifying potential arguments (which is typically performed by an entity extraction system), linking arguments to their event mentions (which is typically performed by an event extraction system), and determining the compatibility between two event arguments (which is provided by an entity coreference resolver), are all non-trivial tasks. In other words, end-to-end event coreference resolution is complicated in part by the fact that an event coreference resolver has to rely on the noisy outputs produced by its upstream components in the standard information extraction pipeline. Many existing event coreference resolvers avoid the hassle of dealing with noisy information and simply adopt a knowledge-lean approach consisting of a pipeline of two components, a trigger de-

tection component that identifies triggers and corresponding subtypes, followed by an event coreference component.

We hypothesize that knowledge-lean approaches are not the right way to go if the ultimate goal is to take event coreference resolvers to the next level of performance. With this in mind, we investigate knowledge-rich approaches in which we derive potentially useful knowledge for event coreference resolution from a variety of sources, including *models* that are trained on tasks that we believe are closely related to event coreference, statistical and linguistic *features* that are directly relevant to the prediction of event coreference links, as well as *constraints* that encode commonsense knowledge of when two event mentions should or should not be coreferent. We start by designing a multi-pass sieve approach that first resolves easy coreference links and then exploits these easy-to-identify coreference links as a source of knowledge to identify difficult coreference links. We then investigate two types of joint models for event coreference resolution, including a joint inference model and a joint learning model, where we encode commonsense knowledge of the inter-dependencies between the various components via hard or soft constraints. In addition, we incorporate non-local information extracted from the broader context preceding an event mention via learning a supervised topic model and modeling discourse salience. Further, we present an unsupervised method for deriving argument compatibility information from a large, unannotated corpus, and develop a transfer-learning framework that transfers the resulting argument (in)compatibility knowledge to an event coreference resolution resolver. Finally, we investigate a multi-tasking neural model that involves simultaneously learning six tasks related to event coreference, and guide the model learning process using cross-task consistency constraints.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Event coreference resolution is a task to determine which event mentions in a document refer to the same real-world event. It is less studied but arguably more challenging. To perform end-to-end event coreference resolution, one has to build an information extraction (IE) pipeline that involves (1) extracting the entity mentions from a given document and determining which of them are coreferent; (2) extracting the event mentions by identifying their trigger words/phrases and determining which entity mentions are their arguments, and a set of attributes that denote, for instance, whether it is a GENERIC or ACTUAL event; and (3) determining which event mentions are coreferent. To better understand the task, consider the following example:

> *Georges Cipriani* $\{left\}_{ev1}$ *a prison in Ensisheim in northern France on parole on Wednesday. He* $\{departed\}_{ev2}$ *the prison in a police vehicle bound for an open prison near Strasbourg.*

In this example, there are two event mentions, *ev1* and *ev2*, which are triggered by the words *left* and *departed* respectively. These event mentions are coreferent because they both refer to the same event of Cipriani leaving the prison.

Intuitively, for two event mentions to be coreferent, not only should they have the same event subtype, but their arguments and attributes should be compatible. In our example, *ev1* and *ev2* have the same subtype, Movement.Transport-Person, thus satisfying the subtype agreement constraint. As far as argument compatibility is concerned, note that an event mention has zero or more *arguments* (the event's participants), each of which plays a role. For instance, *ev1* has three arguments: *Georges Cipriani* is the PERSON argument, *a prison* is the ORIGIN argument, and *Wednesday* is its TIME argument. *ev2* also has three arguments, *He*, *the prison*, and *a police vehicle*, serving as its PERSON, ORIGIN, and INSTRUMENT arguments

respectively. Since the two event mentions have two overlapping roles (i.e., PERSON and ORIGIN) and their arguments are (entity-)coreferent w.r.t. each of these roles, they satisfy the argument compatibility constraint. For attribute compatibility, both *ev1* and *ev2* are ACTUAL event mentions since they actually happened, thus attributes are compatible.

It should be easy to see from this example that event coreference resolution task is challenging. An event coreference resolver has to assume as inputs the noisy outputs of a larger set of upstream components involving entities and events, each of which involves challenging tasks that are far from being solved.

Despite its difficulty, event coreference resolution is the fundamental technology for consolidating the textual information about an event, which is crucial for essentially all high-level natural language processing (NLP) applications. For example, in IE, events and event coreference information have been used for template filling (Humphreys et al., 1997) and automated population of knowledge bases (Ji and Grishman, 2011). In topic detection and tracking, event coreference information is needed to identify new events in a stream of broadcast news stories (Allan et al., 1998). In event-based text summarization, event coreference information has been used to measure the similarity between two events, which in turn can be used to determine whether a sentence is salient or not (Li et al., 2006). Finally, event coreference information has also been used in other applications such as question answering (Narayanan and Harabagiu, 2004) and contradiction detection (De Marneffe et al., 2008).

Our goal is to advance the state of the art in event coreference resolution by exploring knowledge-rich approaches, which can address the error propagation problems in traditional pipeline approach, exploit complex features and incorporate commonsense knowledge into the model via constraints.

First, we develop a easy-first model using six sieves, which makes easy decisions first and subsequently exploit these easy decisions to make hard decision. A sieve is composed of either a set of hand-crafted rules or a machine-learned classifier for classifying a subset

of the mention pairs in a test document. Being an easy-first approach, the six sieves are arranged as a pipeline in decreasing order of precision. When two event mentions are posited as coreferent by a sieve, any argument extracted for one mention will be shared by the other mention. In addition, later sieves can exploit the event coreference decisions made by earlier sieves.

While the above model can propagate the information obtained from early stage to later ones, the error made in the early stage can also propagate to later states. To address the error propagation problem, we develop a joint inference model using Markov Logic Networks (MLNs)(Domingos and Lowd, 2009). In our approach, we jointly perform four key tasks in the IE pipeline: trigger identification and subtyping, argument identification and role determination, entity coreference resolution, and event coreference resolution. The model also exploits inter-dependencies between the various components using both hard and soft MLN formulas.

The joint inference model only allows an event coreference resolver to interact with other components at the inference time. We further investigate a joint learning model which can learn interactions between different components at the model training time. The model simultaneously learns three tasks, including event coreference resolution, trigger detection, and event anaphoricity determination. We build a structured conditional random field model with (1) unary factors, which encode the features specific for each task, and (2) higher-order factors, which capture the interactions between each pair of tasks in a soft manner. However, features used in this model only encode local context. We make two extensions that improve trigger detection by exploiting topic information and improve event coreference by exploiting discourse information. In particular, we propose to train a supervised topic model to infer the topic of each word in a test document, with the goal of understanding each candidate trigger using its global in addition to local context. To exploit discourse information, we introduce a preprocessing component for event coreference resolution where we prune the

candidate antecedents of an event mention that are unlikely to be its correct antecedent based on discourse context.

In addition to non-local information, argument compatibility is frequently incorporated into modern event coreference resolution systems. One of the key challenges in leveraging argument compatibility lies in the paucity of labeled data. To address this problem, we propose a transfer learning framework for event coreference resolution that utilizes a large amount of unlabeled data to learn the argument compatibility between two event mentions. In addition, we adopt an interactive inference network based model to better capture the (in)compatible relations between the context words of two event mentions.

Above mentioned models addressed one or a few aspects of event coreference resolution. The advent of the neural NLP era offers a breakthrough by enabling joint models to scale beyond what has ever been possible. We develop a span-based neural model that involves simultaneously learning six tasks related to event coreference in a multi-task learning framework, and guide the model learning process by incorporating commonsense knowledge into the model that encodes cross-task consistency constraints on event coreference.

1.2 Contribution

In this section, we summarize the major contributions:

- We propose the first multi-pass sieve approach to event coreference resolution. When evaluated on the version of the KBP 2015 corpus available to the participants of EN Task 2 (Event Nugget Detection and Coreference), our approach outperforms the best participating system in KBP 2015.

- We propose a joint inference based event coreference resolver using Markov Logic Networks (MLNs). The model encodes rich NLP features implicitly by augmenting the MLN distribution with low dimensional unit clauses. When evaluated on an English

corpus (KBP 2015) and a Chinese corpus (ACE 2005), our MLN based system achieves statistically significantly better performance than a pipeline-based resolver.

- We propose a joint learning model of event coreference resolution, trigger detection, and event anaphoricity determination. The model encodes features for capturing cross-task interactions. To our knowledge, this is the first attempt to train a mention-ranking model and employ event anaphoricity for event coreference. When evaluated on the KBP 2016 English and Chinese corpora, our model outperforms the independent models.

- We propose two extensions to improve the joint learning model using the *non-local* information provided by a supervised topic model and salient discourse entities.

- We propose a transfer learning framework for event coreference resolution that utilizes a large amount of unlabeled data to learn the argument compatibility between two event mentions and transfers argument (in)compatibility knowledge to the event coreference resolution system.

- We propose a neural model of event coreference resolution that involves simultaneously learning six tasks related to event coreference in a multi-task learning framework, and guide the model learning process by incorporating commonsense knowledge into the model that encodes cross-task consistency constraints on event coreference.

1.3 Outline

The rest of the dissertation is organized as follows. Chapter 2 describes related work on event coreference resolution. In Chapter 3, we describes the multi-pass sieve approach. Chapter 4 introduce the joint inference model. Chapter 5 and 6 introduce the joint learning model and its extensions. Chapter 7 introduce the transfer-learning model. Chapter 8 introduce the constrained multi-task neural model. We conclude with future work in Chapter 9.

CHAPTER 2

RELATED WORK [1]

In this chapter, we discuss the related work on event coreference resolution. While early work on event coreference resolution has employed a rule-based approach (Humphreys et al., 1997), virtually all recent work has adopted a learning-based approach, as described below.

2.1 Supervised Models

2.1.1 Mention-Pair Models

Following early entity coreference resolvers (e.g., Soon et al. (2001), Ng and Cardie (2002a)), many event coreference resolvers adopt a two-step resolution framework. In the first step, a binary classifier (known as a *mention-pair model*) is used to determine whether two event mentions are coreferent. Mention-pair models are typically trained using an off-the-shelf learning algorithm, such as decision trees (Cybulska and Vossen, 2015), maximum entropy (Ahn, 2006; Chen and Ji, 2009), support vector machines (Chen and Ng, 2014), and deep neural networks (Nguyen et al., 2016).

After training, the resulting mention-pair model can be applied to classify the test instances. However, these pairwise classification decisions could violate transitivity, which is an inherent property of the coreference relation. Hence, in the second step, a separate clustering mechanism is needed to coordinate the pairwise decisions and construct a partition. Some researchers employ *agglomerative clustering* algorithms, such as *closest-first* clustering (selecting as the antecedent of an event mention the closest preceding event mention that is classified as coreferent with it by the mention-pair model) and *best-first* clustering (selecting as the antecedent of an event mention the preceding coreferent event mention that has the highest coreference likelihood according to the mention-pair model) (Chen and Ng, 2014;

[1] This chapter was previously published in Lu and Ng (2018).

Peng et al., 2016). Others employ *graph partitioning*. Specifically, given a test document, an undirected weighted graph is first constructed, where the nodes represent the event mentions in the document and the weight of an edge represents the coreference likelihood of the two nodes it connects. Then, a clustering algorithm, such as spectral clustering and divisive clustering, is used to obtain coreference clusters (Chen and Ji, 2009; Chen et al., 2009; Sangeetha and Arock, 2012).

Improvements to this approach include using feature weighting to train a better model (McConky et al., 2012) and training multiple classifiers to handle coreference between event mentions of different syntactic types (Chen et al., 2011).

2.1.2 Generative Models

Though conceptually simple and extensively investigated, mention-pair models and the associated two-step approach suffer from *error propagation*, where errors made by a mention-pair model can propagate to the clustering step. To address this problem, Yang et al. (2015) propose a *supervised* nonparametric generative model for event coreference resolution, building on the framework of the distance-dependent Chinese restaurant process. The model has several key appealing properties. As a clustering model, event mentions are directly assigned to incrementally built coreference clusters. As a nonparametric model, the number of clusters does not need to be known a prior. As a Bayesian model, it can exploit priors, which in this case encode the knowledge provided by a mention-pair model. Finally, being supervised, the model can employ rich features in the modeling process.

2.1.3 Mention-Ranking Models

Recasting event coreference as a classification task may not be a good idea, however. Recall that mention-pair models consider each candidate antecedent of an event mention to be resolved independently of other candidate antecedents. As a result, they can only determine

how good a candidate antecedent is relative to the event mention, but not how good it is relative to other candidate antecedents. Ranking models address this weakness by allowing candidate antecedents of a mention to be ranked *simultaneously*. Motivated by their successful application to entity coreference resolution (Denis and Baldridge, 2008; Durrett and Klein, 2013), Lu and Ng (2017b) train a probabilistic *mention-ranking* model that ranks the candidate antecedents of an event mention so that its correct antecedent has the highest rank. Rather than train a model that maximizes the probability of selecting the correct antecedent for each event mention independently of each other, Lu and Ng train a model to select the antecedents for the event mentions in a document in a *collective* manner by having it assign the highest probability to the correct *vector* of antecedents given all the event mentions. Inference is easy: the most probable (i.e., highest-ranked) candidate antecedent of an event mention is selected to be its antecedent independently of other event mentions.

2.1.4 Easy-First Models

Easy-first models have been successfully applied to many NLP tasks, including entity coreference resolution (Lee et al., 2013). Easy-first coreference models operate in an iterative fashion, aiming to make easy linking decisions first and subsequently exploit these easy decisions (as additional knowledge) to make hard linking decisions.

One of the earliest event coreference resolvers that employs an easy-first approach is Stanford's resolver (Lee et al., 2012). This resolver iteratively bootstraps event coreference output using entity coreference output and vice versa. Specifically, it incrementally builds clusters of event and entity mentions. As clusters become larger, more information becomes available. To exploit the additional information, the features of both the event coreference resolver and the entity coreference model are regenerated.

Liu et al. (2014a) attempt to improve the two-step "classify and cluster" approach described above by adding a third step, where they keep propagating arguments from one

mention in an event coreference cluster to another mention in the same cluster until all mentions in an event coreference cluster share the same arguments. This is an instance of the easy-first approach, as argument propagation helps to identify arguments for event mentions that are otherwise difficult to extract.

Choubey and Huang (2017) build a two-step agglomerative clustering algorithm for within- and cross-document coreference. In the first step, an iterative algorithm that alternates between within- and cross-document event coreference is used to merge within- or cross-document clusters by exploiting the merging decisions made in earlier iterations. Like Liu et al. (2014a), the arguments of the event mentions in the same cluster are shared after each merge. When no more merging can be done, the algorithm proceeds to the second step where additional clusters are merged in an iterative fashion as follows. If the mentions in cluster c_1 are tightly associated (i.e., having the same dependency relations) or loosely associated (i.e., co-occurring in the same sentential context) with those in c_3, and the mentions in cluster c_2 are also tightly or loosely associated with those in c_3, then c_1 and c_2 will be merged.

2.1.5 Joint Models

The aforementioned models all adopt a pipeline architecture, where event triggers and arguments are extracted prior to event coreference resolution. Hence, errors from the upstream components (trigger identification and argument identification) will propagate to the event coreference resolver.

One solution to the error propagation problem is to employ *joint inference* over the outputs of different tasks in the IE pipeline. Chen and Ng (2016) perform joint inference via Integer Linear Programming (ILP) over the outputs of the models trained for the four key tasks in the IE pipeline, namely entity extraction, entity coreference, event extraction, and event coreference. Choubey and Huang (2018) also perform joint inference via ILP by modeling correlations between event coreference chains and document topical structures.

Another solution to the error propagation problem is *joint learning*. Araki and Mitamura (2015) formalize the task of jointly learning event trigger identification and event coreference resolution as a structured prediction problem that is learned using the structured perceptron training algorithm. They employ segment-based decoding with multiple-beam search for event trigger identification, and combine it with best-first clustering for event coreference resolution in document-level joint decoding.

2.2 Semi-Supervised Models

Supervised models suffer from the data acquisition bottleneck, where manually annotating data for all the components in the IE pipeline is expensive. This is especially true for resource-scarce languages. To address this problem, researchers have employed *active learning* to select informative instances, showing that only a small number of training sentences need to be annotated to achieve state-of-the-art event coreference performance (Chen and Ng, 2016). Another attempt is made to utilize large amounts of out of domain text data (Peng et al., 2016). The idea is to (1) represent event structures by five event semantic components, namely action, argument, time, location, and sentence/clause; (2) convert each event component to its corresponding vector representation using different methods, namely explicit semantic analysis, Brown cluster, Word2Vec and dependency-based word embedding; and (3) concatenate all components to form a structured vector representation. These semantic representations are induced from a data set that is not part of the existing annotated event collections and not even from the same domain. Finally, event coreference resolution can be recast as the task of comparing the similarities between event vectors.

2.3 Unsupervised Models

Unsupervised models are proposed to eliminate a model's reliance on annotated data. The vast majority of the existing unsupervised event coreference models are probabilistic gener-

ative models. Bejan and Harabagiu (2014) (B&H) propose several nonparametric Bayesian models for event coreference resolution that probabilistically infer event clusters both *within* a document and *across* multiple documents. One model uses the hierarchical Dirichlet process. It consists of a set of Dirichlet Processes (DPs), in which each DP is associated with each document, and each mixture component is an event coreference cluster shared across documents. This model has the advantage of automatically inferring the number of event clusters in a document. Despite this advantage, the model has limitations in representing feature-rich objects. Consequently, B&H extend this model so that it can consider additional linguistic features derived from WordNet and FrameNet, for instance, instead of just representing each data point by its corresponding word. However, using a feature-rich representation may increase the complexity of a Bayesian model and there is no guarantee that all the features have a positive impact on the task. As a result, B&H extend their model with a feature selection mechanism that automatically selects a finite set of salient features. In addition, they propose another Bayesian model with a mechanism for capturing the structural dependencies between objects.

B&H show that their models that exploit the semantic information extracted from WordNet and FrameNet contribute to coreference performance significantly. However, the lack of comparable lexical knowledge bases complicates the design of event coreference resolvers in languages other than English. To address this problem, Chen and Ng (2015a) design a probabilistic model whose parameters are estimated using EM for computing the probability that two event mentions are coreferent. Its generative process is not language-dependent and does not rely on features extracted from lexical knowledge bases, so it could be applied to languages where neither annotated data nor large-scale knowledge bases are available.

CHAPTER 3

EVENT COREFERENCE RESOLUTION WITH MULTI-PASS SIEVES [1].

Multi-pass sieves were originally applied to entity coreference resolution (Raghunathan et al., 2010; Lee et al., 2013) and have then been successfully applied to many other tasks in natural language processing (NLP) such as temporal relation extraction (Chambers et al., 2014), spatial relation extraction (D'Souza and Ng, 2015b), and disorder mention normalization (D'Souza and Ng, 2015a). Though rarely explicitly mentioned, successful application of a sieve-based approach to a given task depends heavily on the extent to which high-precision rules can be designed for the task. For event coreference resolution, designing high-precision rules is by no means trivial. The reason is that, as mentioned Chapter 1, an event coreference resolver typically assumes as input the noisy outputs of its upstream components. The difficulty in designing high-precision rules makes the successful application of a multi-pass sieve approach to event coreference resolution challenging. In this chapter, we address this challenge, proposing the first multi-pass sieve approach to event coreference resolution.

We first describe the baseline system and then describe the multi-pass sieve model.

3.1 Baseline System

In this section, we describe our baseline system, which operates in two steps. First, it performs event mention detection, which involves detects all explicit mentioning of events with certain specified types in text (Section 3.1.1). Second, it performs event coreference resolution on the event mentions extracted in the first step (Section 3.1.3).

[1]This chapter was previously published in Lu and Ng (2016a)

3.1.1 Event Mention Identification and Subtyping

This component extracts event triggers and determines the semantic type and subtype of each event mention. We recast the task of identifying event triggers as a sequence labeling task, where we train one CRF using the CRF++ package[2]. Each word can trigger multiple event mentions having different types/subtypes, we train one CRF for each type. Specifically, for classifier of type t_j, we create one instance for each word w_i, assigning it a class label that indicates whether it begins a trigger with subtype s_{jk} (B-s_{jk}), is inside a trigger with subtype s_{jk} (I-s_{jk}), begins a trigger with other types (B-$t_{m\neq j}$), is inside a trigger with other types (I-$t_{m\neq j}$) or is outside a trigger (O). So there are (2× number of subtypes of t_j + 2 × number of other types +1) labels in total. Below we describe the features used to represent w_i, which can be divide into three categories: lexical, syntactic and semantic.

Lexical: word unigrams (w_{i-2},w_{i-1},w_i,w_{i+1},w_{i+2}); word bigrams ($w_{i-1}w_i$, w_iw_{i+1}); word trigrams ($w_{i-2}w_{i-1}w_i$, $w_{i-1}w_iw_{i+1}$, $w_iw_{i+1}w_{i+2}$), the part-of-speech tag of w_i; lemmatized word unigrams, bigrams and trigrams.

Syntactic: depth of w_i's node in its syntactic parse tree; the path from the leaf node of w_i to the root in its syntactic parse tree; the phrase structure expanded by the parent of w_i's node; the phrase type of w_i's node.

Semantic: the WordNet synset id of w_i; the WordNet synset ids of the w_i's hypernym, its parent, and its grandparent; When computing these semantic features, we only use the synset corresponding to w_i's first sense.

We improve the recall of event mention detection in a postprocessing process as follows. First, we construct a word list containing triggers that appear infrequently (less than 10 times) in the training data and do not belong to more than one subtype according to the training data. For example, the word "hijack" appears only a few times in the training data

[2]https://taku910.github.io/crfpp/

but is always labeled as "Conflict.Attack". Then, we extract any word as a trigger with the corresponding subtype as long as it appears in the word list.

3.1.2 Event Argument and Role Classification

Event arguments can be entities and argument fillers.[3] Argument fillers corresponds to specific event subtypes, meaning that they will only appear if the corresponding subtype lends itself to such information. Also argument fillers such as Title, Age provide few information for event coreference. Hence we only extract entities as candidate event arguments. To extract entities, we train a CRF (using CRF++) on the texts of LDC2015E29 and LDC2015E68 annotated with Rich ERE entity mentions. The statistics on those corpora are shown in Table 3.1. This classifier jointly identifies and determines the semantic type of each entity mention. Specifically, we create one instance for each word w_i, assigning it a class label that indicates whether it begins an entity mention of type t_j (B-t_j) , is inside an entity of type t_j (I-t_j) or is outside an entity (O). In Rich ERE annotation, entities are labeled by five semantic types: PER, ORG, GPE, LOC, FAC, so there are 11 labels in total. The features used to represent w_i are lexical features including word unigrams, word bigrams, word trigrams formed from words in a window of five, grammatical features namely the part-of-speech tag of w_i, whether w_i is in a NP or not, whether w_i is part of a pronoun, whether the first letter of w_i is in uppercase and semantic features including the WordNet synset id of w_i; the WordNet synset ids of the w_i's hypernym, its parent, and its grandparent.

Then the event argument and role classification component takes as input a set of event mentions from previous component described in Section 3.1.1 and a set of candidate event arguments. For each event mention *em*, it identifies those candidate arguments that are the true arguments of *em* and then assigns a role to each of its true arguments. To implement this

[3]The Rich ERE annotation guideline overview is available from http://cairo.lti.cs.cmu.edu/kbp/2015/event/annotation

Table 3.1. Statistics on LDC2015E29 and LDC2015E68.

		Newswire	Forum
LDC2015E29	Documents	48	43
	Entity mentions	2,751	4,906
LDC2015E68	Documents	–	95
	Entity mentions	–	12,570

component, we jointly learns these two tasks by training a classifier using the $SVM^{multiclass}$. This classifier is also trained on the texts of LDC2015E29 and LDC2015E68 annotated with Rich ERE event arguments. The candidate event arguments include all entity mentions that are extracted using the CRF mentioned above and appear in the same sentence as event trigger. Specifically we create a training instance by pairing each event trigger with each of its candidate arguments. If the candidate argument is indeed a true argument of the trigger, the class label of the training instance is the argument's role. Otherwise, its class label is NONE. There are 27 labels in total, including 26 roles defined in the Rich ERE annotation and NONE. We train this classifier using following 13 features.

Basic: trigger subtype; type of entity mention; head word of entity mention; event subtype + head word; event subtype + entity type; POS of trigger word

Neighbouring words: left/right neighbor word of entity; left/right neighbor word of the entity + word's POS; left/right neighbor word of the trigger + word's POS

Syntactic feature: the phrase structure expanding the parent of trigger in the syntactic parse tree; the phrase type of the trigger; the path from entity to trigger; the dependency path from entity to trigger;

3.1.3 Event Coreference Resolution

This component identifies event coreference links by combining a mention-pair model (Soon et al., 2001), which is a binary classifier that determines whether two event mentions are

co-referring or not, with a best-first single-link clustering algorithm, which selects as the antecedent of an event mention e the best preceding event mention that is classified as coreferent with e. We train a mention-pair model using the libSVM software package (Chang and Lin, 2001). We create positive training instances by pairing each anaphoric event mention em with its closest antecedent and negative training instances by pairing em with each of its preceding event mentions that is not coreferent with em. Each instance is representing using 19 features. We use Stanford CoreNLP package to extract the linguistic information needed to compute these features, including the part-of-speech tags, syntactic parse trees, dependency parse trees and entity coreference chains. As can be seen below, the 19 features can be divided into three groups. For convenience, we use em_2 to refer to an event mention to be resolved and em_1 to refer to a candidate antecedent of em_2.

Group 1 (Event Type and Subtype features). The four features in this group encode: whether em_1 and em_2 agree w.r.t. event type; whether they agree w.r.t. event subtype; the concatenation of their event types; and the concatenation of their event subtypes.

Group 2 (Event Trigger features). The ten features in this group encode: whether em_1 and em_2 have the same trigger; whether they have the same lemmatized trigger; whether the triggers of em_1 and em_2 or the hypernyms of these triggers are in the same synset in WordNet; the concatenation of their triggers; the concatenation of part-of-speech tags of their triggers; whether their triggers agree in number if they are nouns; whether their triggers have the same modifiers if they are nouns; whether their triggers are in the same entity coreference chain if they are nouns; the sentence distance between the triggers of em_1 and em_2; whether this trigger pair appears in the list of trigger pairs of coreferent events in the training set.

Group 3 (Event Argument features).

From the extracted arguments of the event mentions from the event argument component described in Section 3.1.2, we encode seven features: if em_1 and em_2 have arguments of the same role, whether the arguments have the same head word, whether they are in the same

coreference chains, and whether they have the same modifiers; the roles and number of the arguments that only appear in em_1 and the roles and number of the arguments that only appear in em_2

3.2 A Multi-Pass Sieve Approach

In this section, we describe our multi-pass sieve approach to event coreference resolution. The sieve approach has been successfully applied to entity coreference resolution. To our knowledge, ours represents the first attempt to apply the sieve approach to event coreference resolution.

3.2.1 Brief Introduction to Sieves

A sieve is composed of one or more heuristic rules. Each rule extracts a coreference relation between two event mentions. Sieves are ordered by their precision, with the most precise sieve appearing first. To resolve a set of event mentions in a document, the resolver makes multiple passes over them: in the i-th pass, it uses only the rules in the i-th sieve to find an antecedent for each event mention. The candidate antecedents are ordered by their positions in the document. The partial clustering of event mentions generated in the i-th pass is then passed to the i+1-th pass. In this way, later passes can exploit the information computed by previous passes, but the decision make earlier cannot be overridden later.

In our approach, later sieves exploit the decisions made by the earlier sieves as follows. When two event mentions are posited as coreferent by a sieve, any argument extracted for one mention will be shared by the other mention. It is this sharing of argument among coreferent event mentions that will be exploited by the later sieves.

3.2.2 Sieves for Event Coreference

Our sieve approach first extracts event mentions using the same CRF model described in Section 3.1.1, and then employs the following sieves that we designed for event coreference resolution.

1. **Newswire Headline sieve:** this sieve is motivated by the journalistic nature of newswire documents. The first sentence in the newswire documents always contains a detailed explanation of the headline. This sieve posits two event mention in the headline and an event mention the first sentence as coreferent if they have the same subtype and their triggers are in the same WordNet synset.

2. **Strict Event Coreference sieve:** this sieve follows the strict event coreference criteria. Two mentions are posited as coreferent if they satisfy all of the following conditions: (a) they have the same subtypes; (b) their triggers are in the same lemmatized form; (c) at least one of their arguments of the same role are in the same entity coreference chain or are lexically identical (if they are non-pronominal); (d) their triggers are in the same entity coreference chain if they are nouns.

3. **Strict Trigger Match sieve:** this sieve posits two event mentions with noun triggers as coreferent if they have the same subtypes and their triggers have the same lemma and same modifiers.

4. **Semantic Similar Trigger sieve:** this sieve relaxes the Strict Event Coreference Sieve by deleting conditions (b) and (d), but it requires the triggers of the two mentions or the hypernyms of the triggers to be in the same WordNet synset.

5. **Known Coreferent Pair sieve:** this sieve posits two event mentions as coreferent if they have the same subtypes and the trigger pair has appeared in the list of trigger pairs of coreferent event mentions in the training set.

6. **Mention Pair sieve:** this sieve exploits information of mention pairs provided by the baseline system described in the previous section. Specifically, two event mentions are

posited as coreferent with their coreference probability exceeds a certain threshold according to the mention-pair baseline model. The threshold is tuned on development data.

For discussion forum documents, we employ essentially the same sieves except that we replace the first sieve with a sieve that posits two event mentions as coreferent if their triggers and the sentences containing them are identical.This sieve is motivated by the nature of a discussion forum where an author usually quotes a preceding post to which she wants to respond.

3.3 Evaluation

3.3.1 Experiment Setup

Corpus. While different corpora have been used to train and evaluate event coreference resolvers, but as Liu et al. (2014b) pointed out, not all of them were carefully annotated. OntoNotes and ECB have only be partially annotated with event coreference links. Among the publicly-available corpus, the ACE 2005 corpus is arguably the one that is most complete with respect to the annotation of event coreference links. In fact, the majority of recent work on event coreference was evaluated on the ACE 2005 corpus.

As an event coreference corpus, ACE 2005 has a major weakness: it adopts a strict notion of event identity. Specifically, two event mentions were annotated as coreferent if and only if "they had the same agent(s), patient(s), time, and location" (Song et al., 2015), and their event attributes (polarity, modality, genericity, and tense) are not incompatible. This is arguably an overly strict definition of event coreference, as some event mentions are intuitively coreferent even if their time and/or location arguments are not identical.

The KBP 2015 event coreference corpus, which we use to evaluate our model, was created in response to the aforementioned weakness of the ACE 2005 corpus (Song et al., 2015). It was annotated using the Rich ERE guidelines, which are arguably more realistic in the

Table 3.2. Statistics on the official KBP 2015 Event Nugget Detection and Coreference corpus.

Training Data	Newswire	Forum
Documents	81	77
Event mentions	2,219	4,319
Event hoppers	1,461	1874
Evaluation Data	**Newswire**	**Forum**
Documents	98	104
Event mentions	3,788	2,650
Event hoppers	2,440	1,685

sense that they mimic more closely a human's judgment of whether two event mentions are coreferent.

We train event mention identification and subtyping model and event coreference resolution model on the training data of TAC KBP 2015 event detection task. Evaluations is performed on the official evaluation data of TAC KBP 2015 event detection task. The statistics on the dataset are shown in Table 3.2.

Evaluation metrics. To evaluate event coreference performance, we employ four commonly-used coreference scoring measures as implemented in the official scorer of version 1.7 provided by the KBP 2015 organizers namely MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF$_e$ (Luo, 2005) and BLANC (Recasens and Hovy, 2011). [4] Each of these evaluation measures reports results in terms of recall (R), precision (P), and F-score (F). We also report event mention detection performance in terms of recall, precision and F-score, considering a mention correctly detected if it has an exact match with a gold mention in terms of boundary, event type, and event subtype.

Evaluation settings. We evaluate both baseline and proposed system using the predicted event mentions obtained from event detection component described in section 3.1.1.

[4]the official scorer can be found at http://cairo.lti.cs.cmu.edu/kbp/2015/event/scoring

3.3.2 Results and Discussion

In this subsection, we present the results on the official evaluation data. Table 3.3 shows the results of the baseline (row 1) as well as our event coreference resolver (rows 2 to 7) when automatically extracted event mentions are used. Our event mention detection component achieves scores of 50.50% (R), 66.60% (P), and 57.45% (F), which is second best among all participants in KBP 2015 evaluation. The best system achieves score of 58.41% (F). As we can see, the baseline achieves an Avg F-score (average of the F-scores of the four scoring measures) of 37.82%. The subsequent rows show the results when the five sieves are added incrementally. As we can see, our best Avg F-score of 40.32% is obtained when all six sieves are employed. This improvement of 2.5% absolute F-score over the baseline is statistically significant (paired t-test, $p < 0.05$). Our system achieves a better score than the highest score of KBP 2015 evaluation which is 39.65% (F).

A major source of recall error stems from the system's inability to cluster events have few common features. For example, "Somali pirates said Saturday they had received a record nine million dollar ransom in a helicopter air drop for the release of a South Korean supertanker, Samho Dream, with 24 crew. 'The boat was freed this morning agter the payment of nine million dollars to my colleagues,' one of the pirates told AFP by telephone. " In this example, "ransom" and "payment" are triggers of two coreferent events. First just based on the lexical meaning of these two triggers, they share a few information about coreference. Second, in order to know their arguments "a South Korean supertanker" and "the boat" are coreferent, we need more complex analysis.

A major source of precision error stems from the system's tendency to cluster event mentions whose triggers have the same lemma. Although semantic similar trigger sieve are used, more background knowledge is needed to resolve this difficult case.

Table 3.3. Event coreference Results with system event mentions of evaluation data

	MUC			B^3			$CEAF_e$			BLANC			Avg
	R	P	F1	R	P	F1	R	P	F1	R	P	F1	F1
Baseline	29.26	50.78	37.13	39.34	53.88	45.48	35.85	41.66	38.54	23.82	40.93	30.12	37.82
Sieve 1	0.73	73.91	1.45	30.66	66.17	41.91	43.47	36.90	39.92	13.68	58.22	17.13	25.10
+Sieve 2	6.26	53.70	11.22	31.88	65.02	42.78	43.21	38.65	40.80	15.15	49.55	19.92	28.68
+Sieve 3	9.07	56.30	15.63	32.46	64.61	43.21	43.32	39.63	41.39	15.93	50.10	21.29	30.38
+Sieve 4	11.06	52.35	18.27	32.98	63.59	43.44	42.59	40.00	41.25	16.50	48.31	22.22	31.29
+Sieve 5	40.51	48.00	43.93	42.75	48.33	45.37	33.07	46.56	38.67	29.67	39.26	33.11	40.27
+Sieve 6	40.68	48.08	44.07	42.82	48.29	45.39	33.04	46.60	38.67	29.72	39.27	33.14	40.32

3.4 Chapter Summary

In this chapter, we proposed a multi-pass sieve approach to the task of event coreference resolution. When evaluated on the version of the KBP 2015 corpus, our approach achieves an Avg F-score of 40.32%, outperforming the best participating system in TAC KBP 2015 evaluation.

CHAPTER 4

JOINT INFERENCE FOR EVENT COREFERENCE RESOLUTION [1]

As discussed in Chapter 1, pipeline approaches suffer from the error propagation problem. In this chapter, we propose a model based on Markov Logic Networks (MLNs) (Domingos and Lowd, 2009) that jointly performs four key tasks in the IE pipeline, namely, trigger identification and subtyping, argument identification and role determination, entity coreference resolution and event coreference resolution. MLNs are particularly well-suited for modeling *joint inference* tasks in natural language processing (NLP) due to the inherent relational structure and uncertainty typically associated with challenging NLP problems.

Formally, an MLN is a set of pairs (f_i, θ_i) where f_i is a formula in first-order logic and θ_i is a real number. Given a set of constants, an MLN represents a ground Markov network, in which we have one binary random variable for each possible ground atom and one propositional feature for each possible grounding of each first-order formula. The weight associated with the feature is the weight attached to the corresponding formula. The ground Markov network represents the following probability distribution:

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_{f_i} \theta_i N_{f_i}(\omega)\right) \tag{4.1}$$

where $N_{f_i}(\omega)$ is the number of groundings of f_i that evaluate to True given a world ω (an assignment of $\{0, 1\}$ to all ground atoms). The use of first-order logic enables the user to succinctly represent prior, relational knowledge about the application domain, while the weights help model uncertainty in the truth of the first-order logic sentences.

As is commonly known, the major obstacle to the successful application of MLNs to NLP tasks is computational complexity. For event coreference, the rich sets of features that are typically used to model the four IE tasks mentioned above are ill-suited for modeling

[1] This chapter was previously published in Lu et al. (2016).

as explicit MLN formulas since they tend to blow up the size of the MLN due to their high dimensionality, making inference on the MLN infeasible. To address this, we propose a hybrid approach where we embed such features as weighted unit clauses in a low-dimensional space, and then integrate these clauses with the rest of the MLN formulas during inference.

We first describe the baseline system and then describe the joint inference model.

4.1 Baseline System

Our pipeline-based baseline system operates in five steps.

Step 1: Entity extraction. Our entity extraction model jointly identifies the entity mentions and their entity types. We train this model using CRF++[2], treating each sentence as a word sequence. Specifically, we create one instance for each word w and assign it a class label that indicates whether it begins an entity mention with type t_j (B-t_j), is inside an entity mention with type t_j (I-t_j), or is outside an entity mention (O). The features used to represent each instance for training the English CRF and the Chinese CRF are shown in Tables 4.1 and 4.6, respectively.

Step 2: Entity coreference resolution. Our entity coreference classifier is a pairwise classifier that determines whether two entity mentions are coreferent or not. To train this classifier, we employ SVM^{light} (Joachims, 1999), creating training instances using Soon et al.'s (2001) training instance creation method. Each training instance represents two entity mentions in each training document. The class value of a training instance is either positive or negative, depending on whether the two entity mentions are coreferent in the associated text. The features used to represent each instance for training the entity coreference classifiers for English and Chinese are shown in Tables 4.2 and 4.7, respectively.

[2] `https://taku910.github.io/crfpp/`

After training, the resulting classifier can be used to classify each pair of entity mentions extracted in Step 1 as coreferent or not. We select as the antecedent of an entity mention em the closest preceding mention that is classified as coreferent with em.

Step 3: Trigger identification and subtyping. Since ACE allows only single-word triggers, our SVM-based Chinese trigger classifier takes as input a candidate trigger *word* (i.e., a word that survives Li et al.'s (2012) filtering rules) and outputs its event subtype (if it is a true trigger) or *None* (if it is not a trigger). In essence, it jointly (1) identifies event trigger words and (2) assigns a subtype to each identified trigger. To train this classifier, we create one training instance for each word w_i in each training document. If the word does not correspond to a trigger, the class label of the corresponding instance is *None*. Otherwise, the class label is the subtype of the trigger. The features used to represent each instance for training this classifier are shown in Table 4.8.

Because KBP additionally allows multi-word triggers, we recast the task of identifying English triggers as a sequence labeling task, where we train models using CRF++. Recall that since each (multi-)word may trigger multiple event mentions having different (sub)types, we train one CRF for each type. Specifically, to train the CRF for type t_j, we create one instance for each word w_i, assigning it a class label that indicates whether it begins a trigger with subtype s_{jk} (B-s_{jk}), is inside a trigger with subtype s_{jk} (I-s_{jk}), begins a trigger with other types (B-$\mathrm{t}_{m\neq j}$), is inside a trigger with other types (I-$\mathrm{t}_{m\neq j}$) or is outside a trigger (O). The features used to represent each instance for training this CRF are shown in Table 4.3. To improve the recall of event trigger detection, we augment the CRF output with heuristically extracted triggers. Specifically, we first construct a wordlist containing triggers that appear infrequently (less than 10 times) in the training data and do not belong more than one subtype according to the training data. Then, we extract any word as a trigger with the corresponding subtype as long as it appears in the wordlist.

Table 4.1. Features for entity extraction used in the English baseline system. w is the word under consideration.

Lexical	word unigrams, bigrams, and trigrams formed from w with a window size of five.
Grammatical	w's part-of-speech (POS) tag; whether w is part of a NP; whether w is part of a pronoun, whether the first letter of w is in uppercase.
Semantic	the WordNet synset id of w; the WordNet synset ids of w's hypernym, its parent, and its grandparent.

Step 4: Argument identification and role labeling. Our argument identifier and role labeler is a classifier trained using SVM^{light} that jointly learns the tasks of (1) identifying the true arguments of an event mention and (2) assigning a role to each of its true arguments. To train this classifier, we create the training instances by pairing each true event mention *em* (i.e., event mention consisting of a true trigger) with each of *em*'s candidate event arguments, considering an entity mention extracted in Step 1 a candidate argument of *em* if it appears in the same sentence as *em*. If the candidate argument is indeed a true argument of *em*, the class label of the training instance is the argument's role. Otherwise, its class label is *None*. The features used to represent each instance for training the English classifier and the Chinese classifier are shown in Tables 4.4 and 4.9, respectively.

After training, we can apply this classifier to classify test instances. To create test instances, we pair each *candidate* trigger (extracted in Step 3) with each of its candidate event arguments.

Step 5: Event coreference resolution. The event coreference classifier is a pairwise classifier that determines whether two event mentions are coreferent. To train this classifier, we use SVM^{light}, creating training instances using Soon et al.'s(2001) training instance creation method. The features used to represent each instance for training the event coreference classifier for English and Chinese are shown in Tables 4.5 and 4.10, respectively.

Table 4.2. Features for entity coreference resolution used in the English baseline system. en_2 is the entity mention to be resolved and en_1 is a candidate antecedent of en_2.

Lexical	whether en_1 is pronoun; whether en_1 is the subject of the sentence; whether en_1 is noun; whether en_2 is pronoun; whether en_1 is noun; whether en_1 and en_2 have the exactly the same string; whether the modifiers of en_1 and en_2 match; the sentence distance between the strings of $en1$ and en_2.
Grammatical	the number, gender and animacy of en_1 and en_2; whether en_1 and en_2 agree w.r.t. number; whether en_1 and en_2 agree w.r.t. gender; whether en_1 and en_2 agree w.r.t. animacy.

Table 4.3. Features for event trigger identification and subtyping used in the English baseline system. t is the candidate trigger.

Lexical	t's POS tag, lemmatized and unlemmatized word unigrams, word bigrams, word trigrams formed from t with a window size of five.
Syntactic	depth of t in its syntactic parse tree; path from the leaf node of t to the root in its syntactic parse tree; phrase structure expanded by the parent of t's node; phrase type of t's node.
Semantic	WordNet synset id of t; WordNet synset ids of t's hypernym, its parent, and its grandparent.

Table 4.4. Features for event argument identification and role labeling used in the English baseline system. en is a candidate argument of trigger t.

Basic	t's event subtype; en's entity type; en's head word; event subtype + head word; event subtype + entity type; t's POS tag.
Neighboring words	left/right neighbor word of en; left/right neighbor word of en + the word's POS; left/right neighbor word of en + the word's POS.
Syntactic	the phrase structure obtained by expanding the parent of t in the constituent parse tree; the phrase type of t; the path from en to t in the constituent parse tree; the dependency path from en to t.

Table 4.5. Features for event coreference resolution used in the English baseline system. ev_2 is the event mention to be resolved and ev_1 is a candidate antecedent of ev_2.

Event type features	whether ev_1 and ev_2 agree w.r.t. event type; whether they agree w.r.t. event subtype; the concatenation of their event types; and the concatenation of their event subtypes.
Trigger features	whether ev_1 and ev_2 have the same trigger; whether they have the same lemmatized trigger; whether the triggers of ev_1 and ev_2 or the hypernyms of these triggers are in the same WordNet synset; the concatenation of their triggers; the concatenation of POS tags of their triggers; whether their triggers agree in number if they are nouns; whether their triggers have the same modifiers and they are in the same entity coreference chain if they are nouns; the sentence distance between the triggers of ev_1 and ev_2; whether the triggers of ev_1 and ev_2 appear in a training document as a coreferent event mention pair; whether the triggers of ev_1 and ev_2 appear in the first sentence and headline of the document if this is a newswire document; whether the sentence containing the the triggers of ev_1 and ev_2 are identical if this is a discussion forum document.
Argument features	whether ev_1 and ev_2 have arguments of the same role; whether the arguments have the same head word; whether they are in the same coreference chains; whether they have the same modifiers; the roles and number of the arguments that only appear in ev_1; and the roles and number of the arguments that only appear in ev_2.

Table 4.6. Features for entity extraction used in the Chinese baseline system. w is the word under consideration.

Lexical	word unigrams, bigrams, and trigrams formed from w with a window size of five.
Grammatical	w's POS tag; whether w is in a NP; whether w is part of a pronoun.
Wordlist-based	whether w can be found in each of the following 10 wordlists: Chinese surnames; famous GPE and location names (three wordlists); Chinese location suffixes; Chinese GPE suffixes; famous international organization names; famous company names; famous person names; and a list of pronouns.

Table 4.7. Features for entity coreference resolution used in the Chinese baseline system. en_2 is an entity mention to be resolved and en_1 is a candidate antecedent of en_2.

Lexical	whether en_1 is pronoun; whether en_1 is the subject of the sentence; whether en_1 is noun; whether en_2 is pronoun; whether en_1 is noun; whether en_1 and en_2 are the same string; whether the modifiers of en_1 and en_2 match; the sentence distance between $en1$ and en_2;
Grammatical	the number, gender and animacy of en_1 and en_2; whether en_1 and en_2 agree w.r.t. number; whether en_1 and en_2 agree w.r.t. gender; whether en_1 and en_2 agree w.r.t. animacy

Table 4.8. Features for event trigger identification and subtyping used in the Chinese baseline system. t is a candidate trigger.

Lexical	word and POS n-grams formed from t with a window size of three
Syntactic	depth of t in its syntactic parse tree; path from the leaf node of t to the root in its syntactic parse tree; phrase structure expanded by the parent of t's node; the path from the leaf node of t to the governing clause; phrase type of t's node.
Semantic	whether t exists in a predicate list from the Chinese PropBank (Xue and Palmer, 2009); the entry number of t in a Chinese synonym dictionary
Nearest entity information	entity type of the syntactically/physically nearest entity to t in its syntactic parse tree; entity type of the syntactically/physically left/right nearest entity to t in its syntactic parse tree + entity

Table 4.9. Features for event argument identification and role labeling used in the Chinese baseline system. en is a candidate argument of trigger t.

Basic	t's event subtype; en's entity type; en's head word; t's subtype + en's head word; t's event subtype + en's entity type; t's POS tag.
Neighboring words	left/right neighbor word of en; left/right neighbor word of en + the word's POS tag; left/right neighbor word of t + the word's POS tag.
Syntactic	the phrase structure obtained by expanding the parent of t in the constituent parse tree; the phrase type of t; the path from en to t in the constituent parse tree; the dependency path from en to t.

Table 4.10. Features for event coreference resolution used in the Chinese baseline system. ev_2 is the event mention to be resolved and ev_1 is a candidate antecedent of ev_2.

Event type features	whether ev_1 and ev_2 agree w.r.t. event type; whether they agree w.r.t. event subtype; the concatenation of their event types; and the concatenation of their event subtypes.
Trigger features	whether ev_1 and ev_2 have the same trigger; whether the trigger of ev_1 and ev_2 partially matched; whether they have the same lemmatized trigger; the concatenation of their triggers; the concatenation of part-of-speech tags of their triggers; whether their triggers agree in number if they are nouns; whether their triggers have the same modifiers if they are nouns; the sentence distance between the triggers of ev_1 and ev_2; the number of words between ev_1 and ev_2; whether the triggers of ev_1 and ev_2 appear in a training document as a coreferent event mention pair.
Argument features	whether ev_1 and ev_2 have arguments of the same role; whether the arguments have the same head word; whether they are in the same coreference chains; whether they have the same modifiers; the roles and number of the arguments that only appear in ev_1; and the roles and number of the arguments that only appear in ev_2

After training, we apply the resulting classifier to classify test instances. We select as the antecedent of an extracted event mention e the closest preceding mention that is classified as coreferent with e.

4.2 Joint Model

In this section, we describe our MLN-based joint model for event coreference resolution.

4.2.1 MLN Structure

Figure 4.1 shows our proposed MLN for event coreference resolution. It has five predicates subdivided into three categories: query, hidden and evidence.

The *query* predicate `EventCoref`(d,t_1,t_2) is true when two event mentions t_1 and t_2 in document d are coreferent. The *hidden* predicates are those that cannot be directly observed

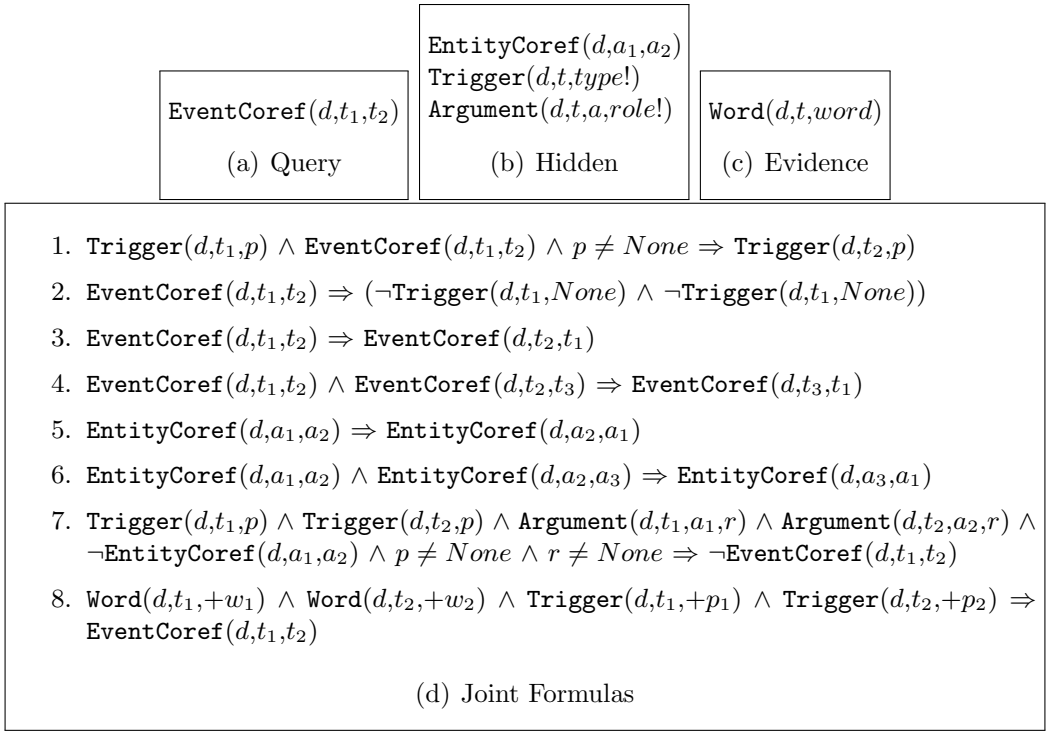

Figure 4.1. MLN structure.

in the data. Our model contains three hidden predicates: (1) Trigger(d,t,p) is true when mention t in document d has event/trigger subtype p. A special type called "None" indicates that t does not contain a trigger. (2) Argument(d,t,a,r) asserts that entity mention a is an argument of event mention t in document d and its role is r. Again, we include a special role called "None", which indicates that the entity mention is not an argument of the event mention. The ! symbol in the predicate definition indicates that every entity mention must take one and only one argument role. (3) EntityCoref(d,a_1,a_2) is true when entity mentions a_1 and a_2 in document d are coreferent. The *evidence* predicates represent (ground) random variables that can be directly observed in the data. In our MLN, we assume that we only observe the words; the predicate Word(d,t,w) is true when mention t in document d equals word w.

[3] http://ir.hit.edu.cn/

The MLN formulas are of two types. The first six formulas have infinite weight which means that they are hard formulas and must be always satisfied. The last two formulas are soft, and their weights will be learned from the data. All logical variables in our formulas are universally quantified and therefore for brevity, we do not use them in the formulas. Formula 1 encodes the hard constraint that if two event mentions are coreferent, then they should share the same trigger subtype. Formula 2 specifies the hard constraint that if event mentions are coreferent, then their triggers subtypes cannot be "None." Formulas 3−6, all of which are hard formulas, specify the commutative and transitive properties of coreferent event and entity mentions. Formula 7, which is a soft formula, specifies the following dependency between the entity mentions that are coreferent and event mentions that are coreferent: for two event mentions t_1 and t_2, if the trigger subtypes of t_1 and t_2 are identical but if there exists a pair of arguments corresponding to t_1 and t_2 respectively that share the same role but the corresponding entity mentions are not coreferent, then event mentions t_1 and t_2 are not coreferent.[4] Formula 8, which is also a soft formula, specifies the dependency between words in the text with the trigger subtypes and event coreference. The + sign in this formula indicates that for every grounding of the variables marked by the + sign, we learn a different weight for the soft formula.

4.2.2 Augmenting the MLN Distribution

Notice that the MLN shown in Figure 4.1 does not model the features used in the baseline systems. These features typically have high dimensionality and encoding them directly in the MLN is quite inefficient. For example, describing a trigram as an MLN formula results in d^3 ground formulas, where d is the number of words in our vocabulary. Therefore, the ground Markov network of an MLN that explicitly models all such high dimensional features would

[4]We restrict the application of Formula 7 to arguments having the following roles: Position, Person, Entity, Organization, Attack, Defendant, Adjudicator, Giver, Agent, Target, and Thing.

be extremely large and infeasible for inference. To address this issue, we implicitly encode the high-dimensional features by embedding them as weighted unit clauses on the groundings of hidden and query predicates in our MLN. Specifically, for each hidden/query ground atom X_i, we derive a single weight, $\phi(X_i)$ using the baseline system. This weight is computed as the distance from the hyperplane for the SVM-based classifiers in the baseline system and as a probability value for the CRF-based classifier in the baseline system. We normalize each weight between the interval [−1,1]. The modified MLN distribution incorporating the new unit clauses is given by

$$P_{\mathcal{M}'}(\omega) \propto \exp\left(\sum_{f_i} \theta_i N_{f_i}(\omega)\right) \Phi(\omega) \tag{4.2}$$

where ω is a world (assignment on every ground atom) and $\Phi(\omega)$ acts as a prior on the set of hidden ($\mathbf{H}$) and query ($\mathbf{Y}$) ground atoms in the original MLN and is given by,

$$\Phi(\omega) = \exp\left(\sum_{X \in \mathbf{H} \cup \mathbf{Y}} \mathbb{I}_X(\omega)\phi(X)\right) \tag{4.3}$$

where $\mathbb{I}_X(\omega)$ is an indicator function that is equal to 1 if X is true in ω and 0 otherwise.

4.2.3 Setting the Soft Formula Weights

During inference time, we dynamically set the weights for the soft formulas (Formulas 7 and 8 in Fig. 4.1) as follows. For each ground soft formula, where the evidence atoms does not make the formula false, we set the formula weight as the summation of SVM weights corresponding to hidden and query atoms in that formula. We then multiply the soft-weights with hyper-parameters η_1 and η_2 for Formula 7 and 8 respectively and tune η_1 and η_2 using a grid-search over values $[0.1, 0.25, 0.5, 0.75, 1.0]$, which optimizes the F1-score of event coreference resolution over the development set.

4.2.4 Inference

Given the prior-augmented MLN, $\mathcal{M}'$, the key task we are interested in is finding a truth assignment to all ground atoms of `EventCoref` that has the maximum probability given

evidence on all ground atoms of `Word`. The following standard MAP inference task, which computes a joint assignment to all hidden and query variables given evidence, can be used to find the desired truth assignment.

$$\arg\max_{\omega} \left\{ \exp\left(\sum_{f_i} \theta_i N_{f_i}(\omega) \right) \Phi(\omega) \right\} \tag{4.4}$$

Unfortunately, the optimization problem given above is NP-hard in general. Moreover, the number of possible worlds in $\mathcal{M}'$ is extremely large and as a result naively searching over this large space in order to solve the optimization problem is computationally infeasible. As a concrete example, for the KBP 15 training dataset, we have 50 million ground atoms.

Fortunately, we can exploit the structure of the MLN given in Figure 4.1 in order to scale up MAP inference. In particular, the subset of ground atoms corresponding to two distinct documents are independent of each other. More formally, let $\mathbf{X}_i$ and $\mathbf{X}_j$ be the subset of ground atoms corresponding to two documents, say D_i and D_j respectively, then $\mathbf{X}_i$ is conditionally independent of $\mathbf{X}_j$ given evidence. Thus, given D documents in our corpus, the joint distribution represented by our MLN can be expressed as a product of D distributions. We can then perform inference independently over each such distribution, which greatly reduces the complexity of inference. Our inference procedure therefore follows an efficient, lazy grounding strategy (Gogate and Domingos, 2011) that grounds the MLN for each document independently and solves Eq. (4.4) for each document separately using Gurobi (2013), a state-of-the-art integer linear programming solver.

4.3 Evaluation

4.3.1 Setup

We perform our evaluation on two corpora, the KBP 2015 English corpus and the Chinese portion of the ACE 2005 training corpus. For English, we train models on 128 of the

training documents, tune parameters (the regularization parameters in SVM classifiers and the weights of the soft MLN formulas) on the remaining 30 training documents, and report results on the official test set.[5] For Chinese, since the ACE 2005 test set is not publicly available, we report five-fold cross validation results on the ACE 2005 training corpus. For each fold experiment, we employ three folds for classifier training, one fold for development (parameter tuning), and one fold for testing.

To evaluate event coreference performance on KBP, we follow the official KBP evaluation and employ four commonly-used scoring measures as implemented in version 1.7 of the official scorer provided by the KBP 2015 organizers, namely MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF_e (Luo, 2005) and BLANC (Recasens and Hovy, 2011), as well as the unweighted average of their F-scores.[6] To evaluate event coreference performance on ACE, we follow previous work on event coreference (e.g., Yang et al. (2015)) and employ the aforementioned four scoring measures as implemented in the latest version (v8) of the CoNLL scorer (Pradhan et al., 2014), as well as the CoNLL score, which is the unweighted average of the MUC, B^3, and CEAF_e F-scores.[7] To our knowledge, there is only one difference between the implementations of the four scoring measures in the two scorers: while the CoNLL scorer considers an event mention correctly detected as long as it has an exact match with a gold event mention in terms of its left and right boundaries, the KBP 2015 scorer is stricter in that it considers an event mention correctly detected by additionally requiring that its event subtype be correctly determined.

[5]Since the KBP 2015 corpus was not annotated with event arguments and entity coreference links, we train our entity mention extractor, our entity coreference resolver, and our event argument identification and role classification model on two LDC corpora provided by the TAC KBP 2015 task organizers (LDC2015E29 and LDC2015E68), as permitted by the rules of the shared task.

[6]The official KBP scorer is available at `http://cairo.lti.cs.cmu.edu/kbp/2015/event/scoring`.

[7]The CoNLL scorer is available at `https://github.com/conll/reference-coreference-scorers`.

Table 4.11. Results for event coreference resolution on KBP 2015 and ACE 2005.

Metric	English/KBP 2015						Chinese/ACE 2005					
	Baseline			**MLNs**			**Baseline**			**MLNs**		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
B^3	53.48	39.21	45.20	50.27	41.63	**45.54**	38.21	37.93	37.66	36.87	42.54	**39.50**
CEAF_e	42.33	38.54	**40.35**	47.53	33.48	39.29	40.28	37.76	38.98	41.02	41.19	**41.10**
MUC	50.52	29.13	36.96	47.07	38.21	**42.18**	40.02	40.27	40.14	39.37	44.70	**41.86**
BLANC	41.16	26.17	32.00	40.61	28.96	**33.30**	24.75	25.67	25.20	22.41	29.07	**25.29**
	Average = 38.64			**Average = 40.08**			CoNLL = 39.02			**CoNLL = 40.82**		

Table 4.12. Results for event trigger identification and subtyping on KBP 2015 and ACE 2005.

English/KBP 2015						Chinese/ACE 2005					
Baseline			**MLNs**			**Baseline**			**MLNs**		
Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
65.05	51.43	57.45	67.97	50.51	**57.95**	67.08	56.44	61.30	66.39	57.37	**61.55**

4.3.2 Results and Discussion

The left half of Table 4.11 shows the results for English event coreference resolution on the KBP 2015 dataset. As can be seen, MLNs outperform the baseline system when evaluated on all but the CEAF_e metrics. W.r.t. the Average metric, MLNs achieve an F-score of 40.38, outperforming the baseline significantly by 1.44 points (paired t-tests, $p < 0.05$). In general, the MLN could detect more event coreference chains than the baseline system, as seen from its higher recall in all but the CEAF_e metrics.[8]

The right half of Table 4.11 shows the results for event coreference resolution on the ACE 2005 Chinese corpus. As can be seen, MLNs outperform the baseline significantly by 1.8 points w.r.t. the CoNLL metric. In fact, MLNs achieve a higher score than the baseline w.r.t. each of the four scoring measures. Similar to what we observed on the KBP corpus,

[8]As is commonly known, CEAF_e sometimes produces unintuitive scores. Specifically, the CEAF_e F-score may drop as more coreference links are correctly identified. See Moosavi and Strube (2016) for a detailed discussion.

the consistently superior performance achieved by the MLN-based resolver can be attributed to its substantially higher recall accompanied by a slightly lower precision. In particular, since MUC is a link-based metric, the fact that the MLNs achieve a higher MUC recall on both datasets suggest that the MLNs are better at discovering event coreference links than the baseline.

One may argue that the MLNs may *not* be better than the baseline at discovering event coreference links: it may simply be the case that the joint inference process has allowed additional triggers to be extracted, which in turn allowed additional event coreference links to be established. To understand whether this is indeed the case, we compute the results for trigger identification and subtyping in Table 4.12. As can be seen, fewer English triggers are extracted after joint inference, whereas the reverse is true for Chinese. These results suggest that at least for English, the higher event coreference recall achieved by the MLNs is not attributable to better trigger identification and subtyping.

A closer examination of the outputs reveals that our resolver is comparatively better at extracting two types of coreference links that are traditionally considered difficult to extract. The first type involves triggers that are lexically different. For example, in the text segment "The former mayor of Detroit, Michigan was sentenced to 28 years in prison ... Prosecutors asked for a minimum of 28 years for Kilpatrick, who resigned from the mayor's office in 2008 ...", the link between event mentions triggered by *former* and *resigned*, both of which have type Personnel.End-position, is discovered by our resolver but not the baseline. The second type involves links between event mentions that are far from each other.

4.4 Chapter Summary

In this chapter, we described a joint inference based event coreference resolver using MLNs. Since encoding rich NLP features in MLNs is a challenging task, we encoded these features implicitly by adding weighted unit clauses to the MLN distribution. Results on an English

corpus (KBP 2015) and a Chinese corpus (ACE 2005) show that our MLN based system achieved statistically significantly better performance than a pipeline-based resolver.

CHAPTER 5

JOINT LEARNING FOR EVENT COREFERENCE RESOLUTION [1]

While the joint inference model can address the error propagation problem, it only allows an event coreference resolver to interact with other components at the inference time. In this chapter, we describe a joint learning model of trigger detection, event coreference, and event anaphoricity. Our choice of these three tasks is motivated in part by their inter-dependencies. It is well-known that trigger detection performance has a huge impact on event coreference performance. Though largely under-investigated, event coreference could also improve trigger detection. For instance, if two event mentions are posited as coreferent, then the underlying triggers must have the same event subtype. While the use of anaphoricity information for entity coreference has been extensively studied (see Ng (2010)), to our knowledge there has thus far been no attempt to explicitly model event anaphoricity for event coreference.[2] Although the mention-ranking model we employ for event coreference also allows an event mention to be posited as non-anaphoric (by resolving it to a *null* candidate antecedent), our decision to train a separate anaphoricity model and integrate it into our joint model is motivated in part by the recent successes of Wiseman et al. (2015a), who showed that there are benefits in jointly training a noun phrase anaphoricity model and a mention-ranking model for entity coreference resolution. Finally, event anaphoricity and trigger detection can also mutually benefit each other. For instance, any verb posited as a non-trigger cannot be anaphoric, and any verb posited as anaphoric must be a trigger. Note that in our joint model, anaphoricity serves as an *auxiliary* task: its intended use is to improve trigger detection and event coreference, potentially mediating the interaction between trigger detection and event coreference.

[1] This chapter was previously published in Lu and Ng (2017a).

[2] Following the entity coreference literature, we overload the term *anaphoricity*, saying that an event mention is anaphoric if it is coreferent with a preceding mention in the associated text.

5.1 Model

5.1.1 Overview

Our model, which is a structured conditional random field, operates at the document level. Specifically, given a test document, we first extract from it (1) all single-word nouns and verbs and (2) all words and phrases that have appeared at least once as a trigger in the training data. We treat each of these extracted nouns and verbs as a candidate event mention. [3] The goal of the model is to make joint predictions for the candidate event mentions in a document. Three predictions will be made for each candidate event mention that correspond to the three tasks in the model: its trigger subtype, its anaphoricity, and its antecedent.

Given this formulation, we define three types of output variables:

- Event subtype variables $\mathbf{t} = (t_1, \ldots, t_n)$. Each t_i takes a value in the set of 18 event subtypes defined in KBP 2016 or NONE, which indicates that the event mention is not a trigger.
- Anaphoricity variables $\mathbf{a} = (a_1, \ldots, a_n)$. Each a_i is either ANAPHORIC or NOT ANAPHORIC.
- Coreference variables $\mathbf{c} = (c_1, \ldots, c_n)$, where $c_i \in \{1, \ldots, i-1, \text{NEW}\}$. In other words, the value of each c_i is the id of its antecedent, which can be one of the preceding event mentions or NEW (if the event mention underlying c_i starts a new cluster).

Each candidate event mention is associated with exactly one coreference variable, one event subtype variable, and one anaphoricity variable. Our model induces the following log-linear probability distribution over these variables:

[3] According to the KBP annotation guidelines, each word may trigger multiple event mentions (e.g., *murder* can trigger two event mentions with subtypes Life.Die and Conflict.Attack). Hence, our treating each extracted word as a candidate event mention effectively prevents a word from triggering multiple event mentions. Rather than complicate model design by relaxing this simplifying assumption, we present an alternative, though partial, solution to this problem wherein we allow each event mention to be associated with multiple event subtypes. See the Appendix for details.

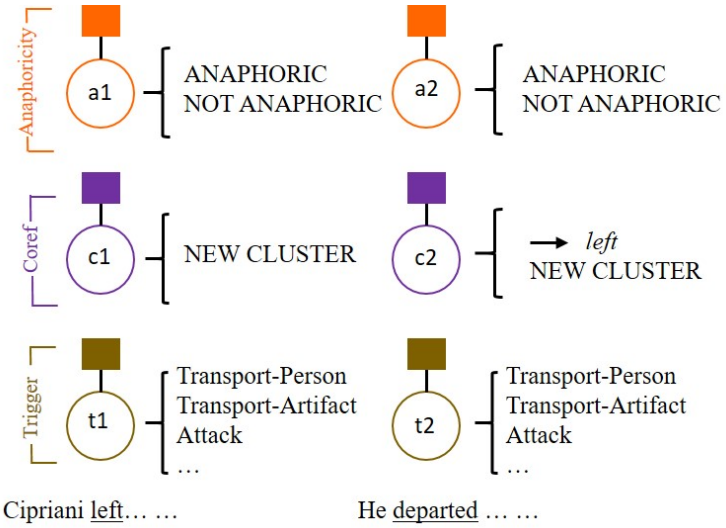

Figure 5.1. Unary factors for the three tasks, the variables they are connected to, and the possible values of the variables. Unary factors encode task-specific features. Each factor is connected to the corresponding output node. The features associated with a factor are used to predict the value of the output node it is connected to when a model is run independently of other models.

$$p(\mathbf{t}, \mathbf{a}, \mathbf{c}|x; \Theta) \propto \exp(\sum_i \theta_i f_i(\mathbf{t}, \mathbf{a}, \mathbf{c}, x)) \tag{5.1}$$

where $\theta_i \in \Theta$ is the weight associated with feature function f_i and x is the input document.

5.1.2 Features

Given that our model is a structured conditional random field, the features can be divided into two types: (1) task-specific features, and (2) cross-task features, which capture the interactions between a pair of tasks. We express these two types of features in factor graph notation. The task-specific features are encoded in unary factors, each of which is connected to the corresponding variable (Figure 5.1). The cross-task features are encoded in binary or ternary factors, each of which couple the output variables from two tasks (Figure 5.2). Next, we describe these two types of features. Each feature is used to train models for both English and Chinese unless otherwise stated.

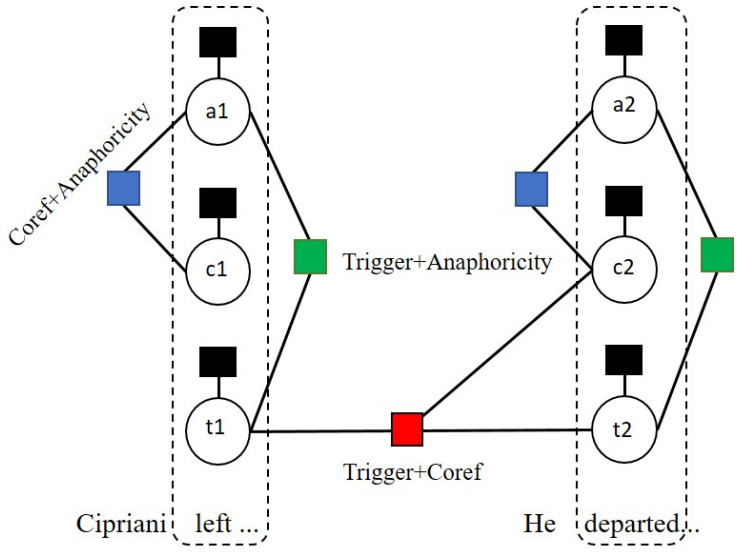

Figure 5.2. Binary and ternary factors. These higher-order factors capture cross-task interactions. The binary anaphoricity and trigger factors encourage anaphoric mentions to be triggers. The binary anaphoricity and coreference factors encourage non-anaphoric mentions to start a NEW coreference cluster. The ternary trigger and coreference factors encourage coreferent mentions to be triggers.

5.1.2.1 Task-Specific Features

We begin by describing the task-specific features, which are encoded in unary factors, as well as each of the three independent models.

Trigger Detection

When applied in isolation, our trigger detection model returns a distribution over possible subtypes given a candidate trigger. Each candidate trigger t is represented using t's word, t's lemma, word bigrams formed with a window size of three from t, as well as feature conjunctions created by pairing t's lemma with each of the following features: the head word of the entity syntactically closest to t, the head word of the entity textually closest to t, the entity type of the entity that is syntactically closest to t, and the entity type of the entity that is textually closest to t.[4] In addition, for event mentions with verb triggers, we use

[4]We train a CRF-based entity extraction model for jointly identifying the entity mentions and their types. Details can be found in Chapter 4

the head words and the entity types of their subjects and objects as features, where the subjects and objects are extracted from the dependency parse trees obtained using Stanford CoreNLP (Manning et al., 2014). For event mentions with noun triggers, we create the same features that we did for verb triggers, except that we replace the subjects and verbs with heuristically extracted agents and patients. Finally, for the Chinese trigger detector, we additionally create two features from each character in t, one encoding the character itself and the other encoding the entry number of the corresponding character in a Chinese synonym dictionary.[5]

Event Coreference

We employ a mention-ranking model for event coreference that selects the most probable antecedent for a mention to be resolved (or NEW if the mention is non-anaphoric) from its set of candidate antecedents. When applied in isolation, the model is trained to maximize the conditional likelihood of collectively resolving the mentions to their correct antecedents in the training texts (Durrett and Klein, 2013). Below we describe the features used to represent the candidate antecedents for the mention to be resolved, m_j.

Features representing the NULL candidate antecedent: Besides m_j's word and m_j's lemma, we employ feature conjunctions given their usefulness in entity coreference (Fernandes et al., 2014). Specifically, we create a conjunction between m_j's lemma and the number of sentences preceding m_j, as well as a conjunction between m_j's lemma and the number of mentions preceding m_j in the document.

Features representing a non-NULL candidate antecedent, m_i: m_i's word, m_i's lemma, whether m_i and m_j have the same lemma, and feature conjunctions including: (1) m_i's word paired with m_j's word, (2) m_i's lemma paired with m_j's lemma, (3) the sentence distance between m_i and m_j paired with m_i's lemma and m_j's lemma, (4) the mention

[5]The dictionary is available from http://ir.hit.edu.cn/. An entry number in this dictionary conceptually resembles a synset id in WordNet.

distance between m_i and m_j paired with m_i's lemma and m_j's lemma, (5) a quadruple consisting of m_i and m_j's subjects and their lemmas, and (6) a quadruple consisting of m_i and m_j's objects and their lemmas.

Anaphoricity Determination

When used in isolation, the anaphoricity model returns the probability that the given event mention is anaphoric. To train the model, we represent each event mention m_j using the following features: (1) the head word of each candidate antecedent paired with m_j's word, (2) whether at least one candidate antecedent has the same lemma as that of m_j, and (3) the probability that m_j is anaphoric in the training data (if m_j never appears in the training data, this probability is set to 0.5).

5.1.2.2 Cross-Task Interaction Features

Cross-task interaction features are associated with the binary and ternary factors.

Trigger Detection and Anaphoricity

We fire features that conjoin each candidate event mention's event subtype, the lemma of its trigger and its anaphoricity.

Trigger Detection and Coreference

We define our joint coreference and trigger detection factors such that the features defined on subtype variables t_i and t_j are fired only if current mention m_j is coreferent with preceding mention m_i. These features are: (1) the pair of m_i and m_j's subtypes; (2) the pair of m_j's subtype and m_i's word; and (3) the pair of m_i's subtype and m_j's word.

Coreference and Anaphoricity

We fire a feature that conjoins event mention m_j's anaphoricity and whether or not a non-NULL antecedent is selected for m_j.

5.1.3 Training

We learn the model parameters Θ from a set of d training documents, where document i contains content x_i, gold trigger annotations t_i^* and gold event coreference partitions C_i^*. Before learning, there are a couple of issues we need to address.

First, we need to derive gold anaphoricity labels a_i^* from C_i^*. This is straightforward: the first mention of each cluster is NOT ANAPHORIC, whereas the rest are ANAPHORIC.

Second, we employ gold event mentions for model training, but training models only on gold mentions is not sufficient: for instance, a trigger detector trained solely on gold mentions will not be able to classify a candidate event mention as NONE during testing. To address this issue, we additionally train the models on candidate event mentions corresponding to non-triggers. We create these candidate event mentions as follows. For each word w that appears as a true trigger at least once in the training data, we create a candidate event mention from each occurrence of w in the training data that is not annotated as a true trigger.

Third, since our model produces event coreference output in the form of an antecedent vector (with one antecedent per event mention), it needs to be trained on antecedent vectors c^*. However, since the coreference annotation for each document i is provided in the form of a clustering C_i^*, we follow previous work on entity coreference resolution (Durrett and Klein, 2013): we sum over all antecedent structures $A(C_i^*)$ that are *consistent* with C_i^* (i.e., the first mention of a cluster has antecedent NEW, whereas each of the subsequent mentions can select any of the preceding mentions in the same cluster as its antecedent).

Next, we learn the model parameters to maximize the following conditional likelihood of the training data with L_1 regularization:

$$L(\Theta) = \sum_{i=1}^{d} \log \sum_{\mathbf{c}^* \in A(C_i^*)} p'(\mathbf{t_i^*}, \mathbf{a_i^*}, \mathbf{c}^* | x; \Theta) + \lambda \|\Theta\|_1 \tag{5.2}$$

where $p'(\mathbf{t}^*, \mathbf{a}^*, \mathbf{c}^*|x;\Theta)$ is $p(\mathbf{t}^*, \mathbf{a}^*, \mathbf{c}^*|x;\Theta)$ augmented with task-specific loss functions, defined as follows:

$$p'(\mathbf{t}^*, \mathbf{a}^*, \mathbf{c}^*|x;\Theta) \propto p(\mathbf{t}^*, \mathbf{a}^*, \mathbf{c}^*) \exp[\alpha_t l_t(\mathbf{t}, \mathbf{t}^*) + \alpha_a l_a(\mathbf{a}, \mathbf{a}^*) + \alpha_c l_c(\mathbf{c}, C^*)] \tag{5.3}$$

where l_t, l_a and l_c are task-specific loss functions, and α_t, α_a and α_c are the associated weight parameters that specify the relative importance of the three tasks in the objective function.

The technique of integrating task-specific loss functions into the objective function, softmax-margin, was introduced by Gimpel and Smith (2010) and used by Durrett and Klein; Durrett and Klein (2013; 2014). By encoding task-specific knowledge, these loss functions can help train a model that places less probability mass on less desirable output configurations.

Our loss function for event coreference, l_c, is motivated by the one Durrett and Klein (2013) developed for entity coreference. It is a weighted sum of the counts of these three error types:

$$l_c(\mathbf{c}, C^*) = \alpha_{c,FA} FA(\mathbf{c}, C^*) + \alpha_{c,FN} FN(\mathbf{c}, C^*) + \alpha_{c,WL} WL(\mathbf{c}, C^*) \tag{5.4}$$

where $FA(\mathbf{c}, C^*)$ is the number of non-anaphoric mentions misclassified as anaphoric, $FN(\mathbf{c}, C^*)$ is the number of anaphoric mentions misclassified as non-anaphoric, and $WL(\mathbf{c}, C^*)$ is the number of incorrectly resolved anaphoric mentions.

Our loss function for trigger detection, l_t, is parameterized in a similar way, having three parameters associated with three error types: $\alpha_{t,FT}$ is associated with the number of non-triggers misclassified as triggers, $\alpha_{t,FN}$ is associated with the number of triggers misclassified as non-triggers, and $\alpha_{t,WL}$ is associated with the number of triggers labeled with the wrong subtype.

Finally, our loss function for anaphoricity determination, l_a, is also similarly parameterized, having two parameters: $\alpha_{a,FA}$ and $\alpha_{a,FN}$ are associated with the number of false anaphors and the number of false non-anaphors, respectively.

Following Durrett and Klein (2014), we use AdaGrad (Duchi et al., 2011) to optimize our objective with $\lambda = 0.001$ in our experiments.

5.1.4 Inference

Inference, which is performed during training and decoding, involves computing the marginals for a variable or a set of variables to which a factor connects. For efficiency, we perform approximate inference using belief propagation rather than exact inference. Given that convergence can typically be reached within five iterations of belief propagation, we employ five iterations in all experiments.

Performing inference using belief propagation on the full factor graph defined in Section 5.1.2 can still be computationally expensive, however. One reason is that the number of ternary factors grows quadratically with the number of event mentions in a document. To improve scalability, we restrict the domains of the coreference variables. Rather than allow the domain of coreference variable c_j to be of size j, we allow a preceding event mention m_i to be a candidate antecedent of m_j if (1) the sentence distance between the two mentions is less than an empirically determined threshold and (2) either they are coreferent at least once in the training data or their head words have the same lemma. Doing so effectively enables us to prune the unlikely candidate antecedents for each event mention. As Durrett and Klein (2014) point out, such pruning has the additional benefit of reducing "the memory footprint and time needed to build a factor graph", as we do not need to create any factor between m_i and m_j and its associated features if m_i is pruned. To further reduce the memory footprint, we additionally restrict the domains of the event subtype variables. Given a candidate event mention created from word w, we allow the domain of its subtype variable to include only NONE as well as those subtypes that w is labeled with at least once in the training data.

For decoding, we employ minimum Bayes risk, which computes the marginals of each variable w.r.t. the joint model and derives the most probable assignment to each variable.

Table 5.1. Results for all three tasks on KBP 2016 evaluation set.

	English						
	MUC	B^3	CEAF_e	BLANC	AVG-F	Trigger	Anaphoricity
KBP2016	26.37	37.49	34.21	22.25	30.08	46.99	-
Indep	22.71	40.72	39.00	22.71	31.28	48.82	27.35
Joint	27.41	40.90	39.00	25.00	33.08	49.30	31.94
Δ	+4.70	+0.18	0.00	+2.29	+1.80	+0.48	+4.59
	Chinese						
	MUC	B^3	CEAF_e	BLANC	AVG-F	Trigger	Anaphoricity
KBP2016	24.27	32.83	30.82	17.80	26.43	40.01	-
Indep	22.68	32.97	29.96	17.74	25.84	39.82	19.31
Joint	27.94	33.01	29.96	20.24	27.79	40.53	23.33
Δ	+5.26	+0.04	0.00	+2.50	+1.95	+0.71	+4.02

5.2 Evaluation

5.2.1 Experimental Setup

We perform training and evaluation on the KBP 2016 English and Chinese corpora. For English, we train models on 509 of the training documents, tune parameters on 139 training documents, and report results on the official KBP 2016 English test set.[6] For Chinese, we train models on 302 of the training documents, tune parameters on 81 training documents, and report results on the official KBP 2016 Chinese test set.

Results of event coreference and trigger detection are obtained using version 1.7.2 of the official scorer provided by the KBP 2016 organizers. To evaluate event coreference performance, the scorer employs four commonly-used scoring measures, namely MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF_e (Luo, 2005) and BLANC (Recasens and Hovy, 2011), as well as the unweighted average of their F-scores (AVG-F). The scorer reports event mention detection performance in terms of F-score, considering a mention correctly detected if it has an exact match with a gold mention in terms of boundary, event type, and

[6]The parameters to be tuned are the α's multiplying the loss functions and those inside the loss functions.

Table 5.2. Results of model ablations on the KBP 2016 evaluation set. Each row of ablation results is obtained by either adding one type of interaction factor to the INDEP. model or deleting one type of interaction factor from the JOINT model. For each column, the results are expressed in terms of changes to the INDEP. model's F-score shown in row 1.

	English			Chinese		
	Coref	Trigger	Anaph	Coref	Trigger	Anaph
INDEP.	31.28	48.82	27.35	25.84	39.82	19.31
INDEP.+CorefTrigger	+0.39	+0.42	−0.05	+0.95	+0.56	−0.37
INDEP.+CorefAnaph	+0.40	−0.08	+3.45	+0.37	+0.04	−0.11
INDEP.+TriggerAnaph	+0.11	+0.38	+1.35	+0.14	+0.52	+0.02
JOINT−CorefTrigger	+0.56	−0.06	+4.41	+0.19	+0.35	+3.34
JOINT−CorefAnaph	+0.63	+0.66	+1.46	+1.50	+0.88	+0.28
JOINT−TriggerAnaph	+1.89	+0.50	+4.01	+1.65	+0.50	+1.79
JOINT	+1.80	+0.48	+4.59	+1.95	+0.71	+4.02

event subtype. In addition, we report anaphoricity determination performance in terms of the F-score computed over anaphoric mentions, counting a mention as a true positive if it has an exact match with an anaphoric gold mention in terms of boundary.

5.2.2 Results and Discussion

Results are shown in Table 5.1 where performance on all three tasks (event coreference, trigger detection, and anaphoricity determination) is expressed in terms of F-score. The top half of the table shows the results on the English evaluation set. Specifically, row 1 shows the performance of the best event coreference system participating in KBP 2016 (Lu and Ng, 2016b). This system adopts a pipeline architecture. It first uses an ensemble of one-nearest-neighbor classifiers for trigger detection. Using the extracted triggers, it then applies a pipeline of three sieves, each of which is a 1-nearest-neighbor classifier, for event coreference. As we can see, this system achieves an AVG-F of 30.08 for event coreference and an F-score of 46.99 for trigger detection.

Row 2 shows the performance of the independent models, each of which is trained independently of the other models. Specifically, each independent model is trained using only

the unary factors associated with it. As we can see, the independent models outperform the top KBP 2016 system by 1.2 points in AVG-F for event coreference and 1.83 points for trigger detection.

Results of our joint model are shown in row 3. The absolute performance differences between the joint model and the independent models are shown in row 4. As we can see, the joint model outperforms the independent models for all three tasks: by 1.80 points for event coreference, 0.48 points for trigger detection and 4.59 points for anaphoricity determination. Most encouragingly, the joint model outperforms the top KBP 2016 system for both event coreference and trigger detection. For event coreference, it outperforms the top KBP system w.r.t. all scoring metrics, yielding an improvement of 3 points in AVG-F. For trigger detection, it outperforms the top KBP system by 2.31 points.

The bottom half of Table 5.1 shows the results on the Chinese evaluation set. The top KBP 2016 event coreference system on Chinese is also the Lu and Ng (2016b) system. While the top KBP system outperforms the independent models for both tasks (by 0.59 points in AVG-F for event coreference and by 0.19 points for trigger detection), our joint model outperforms the independent models for all three tasks: by 1.95 points for event coreference, 4.02 points for anaphoricity determination, and 0.71 points for trigger detection. Like its English counterpart, the joint model for Chinese outperforms the top KBP system for both event coreference and trigger detection. For event coreference, it outperforms the top KBP system w.r.t. all scoring metrics, yielding an absolute improvement of 1.36 points in AVG-F. For trigger detection, it outperforms the top KBP system by 0.52 points.

For both datasets, the joint model's superior performance to the independent coreference model stems primarily from considerable improvements in MUC F-score. As MUC is a link-based measure, these results provide suggestive evidence that joint modeling has enabled more event coreference links to be discovered.

5.2.3 Model Ablations

To evaluate the importance of each of the three types of joint factors in the joint model, we perform ablation experiments.[7] Table 5.2 shows the results on the English and Chinese datasets when we add each type of joint factors to the independent model and remove each type of joint factors from the full joint model. The results of each task are expressed in terms of changes to the corresponding independent model's F-score.

Coref-Trigger interactions. Among the three types of factors, this one contributes the most to coreference performance, regardless of whether it is applied in isolation or in combination with the other two types of factors to the independent coreference model. In addition, it is the most effective type of factors for improving trigger detection. When applied in combination, it also improves anaphoricity determination, although less effectively than the other two types of factors.

Coref-Anaphoricity interactions. When applied in isolation to the independent models, this type of factors improves coreference resolution but has a mixed impact on anaphoricity determination. When applied in combination with other types of factors, it improves both tasks, particularly anaphoricity determination. Its impact on trigger detection, however, is generally negative.

Trigger-Anaphoricity interactions. When applied in isolation to the independent models, this type of factors improves both trigger detection and anaphoricity determination. When applied in combination with other types of factors, it still improves anaphoricity determination (particularly on English), but has a mixed effect on trigger detection. Among the three types of factors, it has the least impact on coreference resolution.

[7]Chen and Ng (2013) also performed ablation on their ACE-style Chinese event coreference resolver. However, given the differences in the tasks involved (e.g., they did not model event anaphoricity, but included tasks such as event argument extraction and role classification, entity coreference, and event mention attribute value computation) and the ablation setup (e.g., they ablated individual tasks/components in their pipeline-based system in an incremental fashion, whereas we ablated interaction factors rather than tasks), a direct comparison of their observations and ours is difficult.

5.2.4 Error Analysis

Next, we conduct an analysis of the major sources of error made by our joint coreference model.

5.2.4.1 Two Major Types of Precision Error

Erroneous and mistyped triggers. Our trigger model tends to assign the same subtype to event mentions triggered by the same word. As a result, it often assigns the wrong subtype to triggers that possess different subtypes in different contexts. For the same reason, words that are sometimes used as triggers and sometimes not are often wrongly posited as triggers when they are not. These two types of triggers have in turn led to the establishment of incorrect coreference links.[8]

Failure to extract arguments. In the absence of an annotated corpus for training an argument classifier, we exploit dependency relations for argument extraction. Doing so proves inadequate, particularly for noun triggers, owing to the absence of dependency relations that can be used to reliably extract their arguments. Moreover, using dependency relations does not allow the extraction of arguments that do not appear in the same sentence as their trigger. Since the presence of incompatible arguments is an important indicator of non-coreference, our model's failure to extract arguments has resulted in incorrect coreference links.

[8]In our joint model, mentions that are posited as coreferent are encouraged to have the same subtype. While it can potentially fix the errors involving coreferent mentions that have different subtypes, it cannot fix the errors in which the two mentions involved have the same erroneous subtype.

5.2.4.2 Three Major Types of Recall Error

Missing triggers. Our trigger model fails to identify trigger words/phrases that are unseen or rarely-occurring in the training data. As a result, many coreference links cannot be established.

Lack of entity coreference information. Entity coreference information is useful for event coreference because the corresponding arguments of two event mentions are typically coreferent. Since our model does not exploit entity coreference information, it treats two lexically different event arguments as non-coreferent/unrelated, which has in turn weakened its ability to determine two event mentions as coreferent. This issue is particularly serious in discussion forum documents, where it is not uncommon to see pronouns serve as subjects and objects of event mentions. The situation is further aggravated in Chinese documents, where zero pronouns are prevalent.

Lack of contextual understanding. Our model extracted only features from the sentence in which an event mention appeared, but additional contextual information present in neighboring sentences may be needed for correct coreference resolution. This is particularly true for discussion forum documents, where the same event was described differently by different people. For example, when describing the fact that Tim Cook will attend the Apple's Istanbul Store opening, one person said "Cook is expected to return to Turkey for the store opening", and another person described this event as "Tim travels abroad YET AGAIN to be feted by the not-so-high-and-mighty". It is by no means easy to determine that *return* and *travel* trigger two coreferent event mentions in these sentences.

5.3 Chapter Summary

In this chapter, we describe a joint model of event coreference resolution, trigger detection, and event anaphoricity determination. The model is novel in its choice of tasks and the

cross-task interaction features. When evaluated on the KBP 2016 English and Chinese corpora, our model illustrates the effectiveness of joint learning model and outperforms the independent model.

CHAPTER 6

JOINT LEARNING FOR EVENT COREFERENCE RESOLUTION WITH NON-LOCAL INFORMATION [1]

We hypothesize that the power of this joint event coreference model has not been fully exploited and seek to extend it in this chapter. Our extensions are based on the observation that the strength of a joint model stems from its ability to facilitate cross-task knowledge transfer. In other words, the better we can model *each* task involved, the more we can potentially get out of joint modeling. Given this observation, we seek to improve the modeling of these tasks in this joint model as follows.

First, we improve trigger detection by exploiting topic information. State-of-the-art trigger detectors, including those based on deep neural networks (e.g., Nguyen et al. (2016)), classify each candidate trigger using local information and largely ignore the fact that the *topic* of the document in which a trigger appears plays an important role in determining its event subtype. To understand the usefulness of document topics, consider the examples in Table 6.1: although all five events have similar trigger words, we can see that the meaning of the triggers and their event subtypes are different in different contexts. Hence, if an event coreference model knows that the topics of these two documents are different, it can exploit this information to more accurately classify their event subtypes. In particular, we propose to train a supervised topic model to infer the topic of each word in a test document, with the goal of understanding each candidate trigger using its *global* in addition to local context.

Second, we improve event coreference by exploiting *discourse* information. Specifically, we introduce a *preprocessing* component for event coreference resolution where we *prune* the candidate antecedents of an event mention that are unlikely to be its correct antecedent based on *discourse context*. In essence, this discourse-based preprocessing step seeks to simplify the

[1] This chapter was previously published in Lu and Ng (2020).

Table 6.1. Event coreference resolution examples.

Three journalists at The New York Times on Tuesday announced plans to {leave}$_{ev1}$ the newspaper. The {departures}$_{ev2}$ follow moves last month by several other Times employees, all of whom were {leaving}$_{ev3}$ to join digital companies.
Pakistan's Interior Ministry has ordered New York Times Reporter to {leave}$_{ev4}$. The ministry gave no explanation for the expulsion order. "You are therefore advised to {leave}$_{ev5}$ the country within 72 hours," the order stated.

job of the event coreference model by reducing the number of candidate antecedents it has to consider for a given event mention. We encode the discourse context of an event mention using the entities that are *salient* at the point of the discourse in which the event mention appears. To our knowledge, we are the first to show that event coreference performance can be improved using discourse contexts that are encoded using salient discourse entities.

6.1 Model

Following the model proposed in Chapter 5, we employ a structured conditional random field, which operates at the document level. Specifically, given a test document, we first extract from it all single- and multi-word nouns and verbs that have appeared at least once as a trigger in the training data. We treat each of these extracted nouns and verbs as a candidate event mention. The goal of the model is to make joint predictions for the candidate event mentions in a document. Three predictions will be made for each candidate event mention that correspond to the three tasks in the model: its trigger subtype, its induced topic, and its antecedent.

Given this formulation, we define three types of output variables. The first type consists of event subtype variables $\mathbf{s} = (s_1, \ldots, s_n)$. Each s_i takes a value in the set of the 18 event subtypes defined in KBP 2017 or NONE, which indicates that the event mention is not a trigger. The second type consists of coreference variables $\mathbf{c} = (c_1, \ldots, c_n)$, where $c_i \in \{1, \ldots, i-1, \text{NEW}\}$. In other words, the value of each c_i is the id of its antecedent, which can

be one of the preceding event mentions, or NEW (if the mention underlying c_i starts a new cluster). The third type consists of topic variables $\mathbf{t} = (t_1, \ldots, t_n)$. Each t_i takes a value in a 19-element set in which the topics have a one-to-one correspondence with the event subtype labels defined above. Despite this one-to-one mapping, these two types of labels should not be interpreted in the same manner. As we will see, a word's induced topic label is influenced by our supervised topic model, whereas a word's subtype is not.

Each candidate event mention is associated with one coreference variable, one event subtype variable, and one topic variable. Our model induces a probability distribution over these variables:

$$p(\mathbf{s}, \mathbf{c}, \mathbf{t}|x; \Theta) \propto \exp(\sum_i \theta_i f_i(\mathbf{s}, \mathbf{c}, \mathbf{t}, x))$$

where $\theta_i \in \Theta$ is the weight associated with feature function f_i and x is the input document.

6.1.1 Independent Models

Trigger Detection Model

Each instance for training the trigger detection model corresponds to a candidate trigger in the training set, which is created as follows. For each word w that appears as a true trigger at least once in the training data, we create a candidate trigger from each occurrence of w in the training data. If a given occurrence of w is a true trigger in the associated document, the class label of the corresponding training instance is its subtype label. Otherwise, we label the instance as NONE.

Each candidate trigger m is represented using features generated from the following feature templates: m's word, m's lemma, word bigrams formed with a window size of three from m; feature conjunctions created by pairing m's lemma with each of the following features: the head word of the entity syntactically closest to m, the head word of the entity textually

closest to m, the entity type of the entity that is syntactically closest to m, and the entity type of the entity that is textually closest to m.[2] In addition, for event mentions with verb triggers, we use the head words and the entity types of their subjects and objects as features, where the subjects and objects are extracted from the dependency parses produced by Stanford CoreNLP (Manning et al., 2014). For event mentions with noun triggers, we essentially create the same features except that we replace the subjects and verbs with heuristically extracted agents and patients.

Topic Model

Our first extension to the model in Chapter 5 seeks to improve trigger detection using topic information. We train a supervised topic model to infer the topic of each word in a test document, with the goal of understanding each candidate trigger using its *global* in addition to local context.

Like the trigger detection model, each training instance corresponds to a candidate trigger. The class label is the topic label of the candidate trigger. We have 19 topic labels in total: there is a one-to-one correspondence between the 18 subtype labels and 18 of the topic labels. The remaining topic label is OTHER, which is reserved for those words that do not belong to any of the 18 topics. Topic labels can be derived directly from subtype labels given the one-to-one correspondence between them. Each candidate trigger is represented using 19 features, which correspond to the 19 topic labels. The value of a feature, which is derived from the output of a LabeledLDA model (Ramage et al., 2009), encodes the probability that the candidate trigger belongs to the corresponding topic.

To train the LabeledLDA model, we first apply LabeledLDA using the Mallet toolkit (McCallum, 2002) to the training documents, which learns a distribution over words for

[2]We use an in-house CRF-based entity extraction model to jointly identify the entity mentions and their types.

each topic, β. We represent each training document using the candidate triggers as well as the context words that are useful for distinguishing the topics.[3] To get the useful context words, we rank the words in the training documents by their weighted log-likelihood ratios:

$$P(w_i|m_j, v_k) \log \frac{P(w_i|m_j, v_k)}{P(w_i|m_j, \neg v_k)}$$

where w_i, m_j and v_k denote the ith word in the vocabulary, the jth candidate trigger word and the kth subtype (including NONE), respectively. Intuitively, a word w_i will have a high rank with respect to a candidate trigger word m_j of subtype v_k if it appears frequently with m_j of subtype v_k and infrequently with m_j of other subtypes. We employ as the useful context words the top 125 words ranked by the weighted log likelihood ratio w.r.t. each pair of trigger and subtype. The label set of each training document is the set of subtypes collected from all the triggers in the document plus NONE.

After training, we apply the resulting LabeledLDA model to a test document, which is represented using the candidate triggers and the useful context words, as defined above. Specifically, given a test document, we (1) apply the model to infer the distribution of topics in the document, and then (2) compute the posterior distribution of topics given each candidate trigger in the document using Bayes rule as follows:

$$P(z|m) \propto P(m|z : \beta)P(z)$$

where $P(z)$ is the distribution of topic z in the test document, $P(m|z : \beta)$ is the topic-dependent distribution of candidate triggers m that is learned from the training documents, and $P(z|m)$ is the posterior distribution of z given m in the test document. We use this

[3]If a candidate trigger is a multi-word phrase, we treat it as a "word" by concatenating its constituent words using underscores (e.g.,"step down" is represented as "step_down").

posterior distribution to generate features for representing each instance for training/testing the topic model, as described above.

Note that while the label sets used by the trigger detector and the topic model are functionally equivalent, they are trained using different feature sets. The features used by the trigger detector encodes a candidate trigger's local context, while the features used by the topic model encodes its global context (e.g., its relationship with other words).

Event Coreference Model

Our event coreference model is an adaptation of Durrett and Klein's (2013) mention-ranking model, which was originally developed for entity coreference, to the task of event coreference. This model selects the most probable antecedent for a mention to be resolved from its set of candidate antecedents (or NEW if the mention is non-anaphoric).

We employ two types of feature templates to represent the candidate antecedents for the event mention to be resolved, m_j. The first type is composed of features that represent the NULL candidate antecedent. These include: m_j's word, m_j's lemma, a conjoined feature created by pairing m_j's lemma with the number of sentences preceding m_j, and another conjoined feature created by pairing m_j's lemma with the number of mentions preceding m_j in the document. The second type is composed of features that represent a non-NULL candidate antecedent, m_i. These include m_i's word, m_i's lemma, whether m_i and m_j have the same lemma, and the following feature conjunctions: (1) m_i's word paired with m_j's word, (2) m_i's lemma paired with m_j's lemma, (3) the sentence distance between m_i and m_j paired with m_i's lemma and m_j's lemma, (4) the mention distance between m_i and m_j paired with m_i's lemma and m_j's lemma, (5) a quadruple consisting of m_i and m_j's subjects and their lemmas, and (6) a quadruple consisting of m_i and m_j's objects and their lemmas.

Our second extension to model in Chapter 5 involves leveraging *discourse* information to improve this event coreference model. Specifically, we introduce a *preprocessing* component

for event coreference resolution where we *prune* the candidate antecedents of an event mention that are unlikely to be its correct antecedent based on *discourse context*. The idea is to (1) encode the discourse context of each event mention in a document using the *entities* that are *salient* at the point of the discourse in which the event mention appears, and by hypothesizing that two event mentions that appear in different discourse contexts are unlikely to be coreferent, we (2) prune any candidate antecedent of an event mention m whose discourse context is different from that of m, allowing the event coreference model to resolve an event mention to one of the candidate antecedents that survive this discourse-based filtering step. In essence, this preprocessing step seeks to simplify the job of the event coreference model by reducing the number of candidate antecedents it has to consider for a given event mention.

Since we aim to encode the discourse context of each event mention using the entities that are salient at the point of the discourse in which the event mention appears, we need to compute the salience score of each entity E w.r.t. each event mention m. We employ the following formula, which was proposed by Chen and Ng (2015b):

$$\sum_{e \in E} g(e) \times decay(e)$$

In this formula, e is a mention of entity E that appears in either the same sentence as m or one of its preceding sentences. $g(e)$ is a score that is computed based on the grammatical role of e in the sentence: 4 if e is a subject, 2 if it is an object, and 1 otherwise. $decay(e)$ is a decay factor that is set to 0.5^{dis}, where dis is the sentence distance between e and m. We compute discourse entities using Stanford CoreNLP's neural entity coreference resolver and grammatical roles using CoreNLP's syntactic dependency parser.

Next, we define the discourse context of an event mention m to be the list of entities whose salience score is at least 1 when computed w.r.t. m. As noted before, we aim to prune the unlikely candidate antecedents of an event mention m, namely those candidates

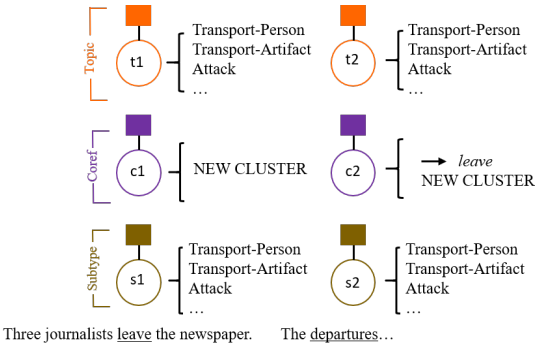

Figure 6.1. Unary factors for the three tasks, the variables they are connected to, and the possible values of the variables.

whose discourse contexts are different from that of m. Rather than heuristically defining a function for computing the similarity between two different discourse contexts, we train a ranker that ranks the candidate antecedents of m based on two types of features derived from their discourse contexts:

Salience score ratios (SSRs): For each entity E that appears in the discourse contexts of both candidate antecedent c and m, we first compute E's SSR as the ratio of E's salience score computed w.r.t. m to E's salience score computed w.r.t. c. (If this ratio is less than 1, we take its reciprocal.) Then, for each (c, m) pair, we create five features that encode the number of entities whose SSR falls into each of these five intervals: [1,1], (1, 2], (2, 3], (3,4], (4,5], and [5, inf]. Intuitively, c's and m's discourse contexts tend to be more similar if they have more entities in the lower buckets.

Lexical features: For each mention em_1 of each entity in candidate antecedent c's discourse context and each mention em_2 of each entity in m's discourse context, we create a lexical feature that pairs em_1's head with em_2's head.

To train this ranker, we employ the same log-linear model as the one used for the event coreference model, where the training objective is to maximize the likelihood of selecting the correct antecedent for each event mention.

After training, we apply this ranker to prune all but the top k candidate antecedents of each event mention in a test document. These k candidate antecedents, together with the NULL candidate antecedent, will be ranked by the event coreference model, and the highest-ranked candidate will be selected as the antecedent of the event mention under consideration.[4] We treat k as a hyperparameter and tune it on the development set.

It is worth noting that we prune the candidate antecedents of the event mentions not only in the test set but also in the training set. We produce the top k candidate antecedents of each event mention in the training set via five-fold cross-validation over the training documents.

Figure 6.1 illustrates the unary factors, which encode the features used in the three independent models. Specifically, the sentence fragment at the bottom of the figure contains two event mentions, one triggered by *leave* and the other by *departure*. Each of them is associated with three variables, one for each of the three models. Next to each variable is the set of possible values of that variable.

6.1.2 Joint Learning

To perform joint training over the three models described in the previous subsection, we need to define (1) features that capture the interaction between the two tasks, (2) the joint training scheme, and (3) the inference mechanism.

Cross-Task Interaction Features

Our cross-task interaction features, which capture the *pairwise* interaction between our tasks, are associated with ternary factors, as described below.

[4]The discourse preprocessing module does not handle NULL candidate antecedents, so they will always be available to the event coreference model.

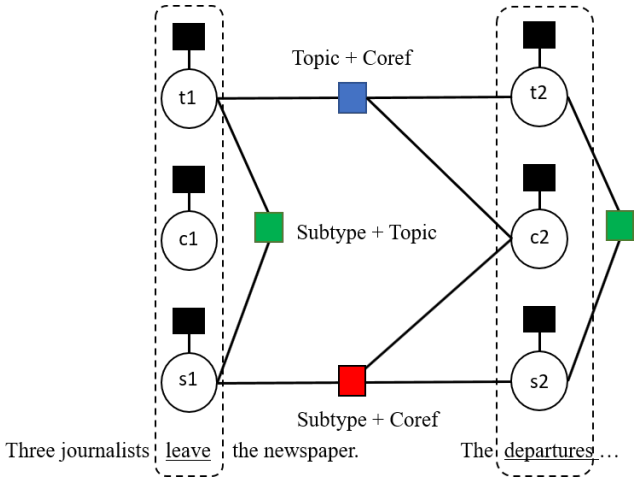

Figure 6.2. Binary and ternary factors.

Trigger detection and coreference. We define our joint coreference and trigger detection factors such that the features defined on subtype variables s_i and s_j are fired only if current mention m_j is coreferent with preceding mention m_i. These features are: (1) the pair of m_i and m_j's subtypes; (2) the pair of m_j's subtype and m_i's word; and (3) the pair of m_i's subtype and m_j's word.

Trigger detection and topic modeling. We fire features (encoded as binary factors) that conjoin each candidate event mention's event subtype, its topic and the lemma of its trigger.

Topic modeling and coreference. Our joint coreference and topic modeling factors and features are the same as those for trigger detection and coreference, except that event subtype labels are replaced with topic labels. In other words, the features are defined on the topic labels.

Figure 6.2 shows the cross-task interaction features. The green factor is binary, connecting a subtype variable and a topic variable. The red factor is ternary, connecting two subtype variables to a coreference variable. Finally, the blue factor is also ternary, connecting topic with coreference.

Training

The joint training scheme seeks to learn the model parameters Θ from a set of d training documents, where document i contains content x_i, gold trigger annotations $\mathbf{s_i^*}$, inferred topic labels $\mathbf{t_i^*}$ from LabeledLDA model using Gibbs sampling and gold event coreference partition C_i^*, by maximizing the conditional likelihood of the training data with L_1 regularization:[5]

$$L(\Theta) = \sum_{i=1}^{d} \log \sum_{\mathbf{c}^* \in A(C_i^*)} p'(\mathbf{s_i^*}, \mathbf{t_i^*}, \mathbf{c}^* | x_i; \Theta) + \lambda \|\Theta\|_1$$

where $p'(\mathbf{s}^*, \mathbf{t_i^*}, \mathbf{c}^* | x; \Theta)$ is $p(\mathbf{s}^*, \mathbf{t_i^*}, \mathbf{c}^* | x; \Theta)$ augmented with task-specific loss functions. Specifically,

$$p'(\mathbf{s}^*, \mathbf{t_i^*}, \mathbf{c}^* | x; \Theta) \propto p(\mathbf{s}^*, \mathbf{t_i^*}, \mathbf{c}^* | x; \Theta) \exp[\alpha_s l_s(\mathbf{s}, \mathbf{s}^*) + \alpha_t l_t(\mathbf{t}, \mathbf{t}^*) + \alpha_c l_c(\mathbf{c}, C^*)]$$

where l_s, l_t and l_c are task-specific loss functions[6], and α_s, α_t and α_c are the associated weight parameters that specify the relative importance of the two tasks in the objective function. [7] We use AdaGrad (Duchi et al., 2011) for optimization with $\lambda = 0.001$.

---

[5] In the conditional log likelihood function, $A(C_i^*)$ is the set of antecedent structures that are consistent with C_i^*. Since our model needs to be trained on antecedent vectors $\mathbf{c}^*$ but the gold coreference annotation for each document i is provided in the form of a clustering C_i^*, we need to sum over all consistent antecedent structures.

[6] The loss function for event coreference, which is introduced by Durrett and Klein (2013) for entity coreference resolution, is a weighted sum of (1) the number of anaphoric mentions misclassified as non-anaphoric, (2) the number of non-anaphoric mentions misclassified as anaphoric, and (3) the number of incorrectly resolved mentions. The loss function for trigger detection is parameterized in a similar way, having three parameters associated with (1) the number of non-triggers misclassified as triggers, (2) the number of triggers misclassified as non-triggers, and (3) the number of triggers labeled with the wrong subtype. The loss function for topic detection is defined in similar way as trigger detection.

[7] These weight parameters, as well as those that are used within the loss functions, are tuned on the development set using grid search.

Inference

Inference, which is performed during training and decoding, involves computing the marginals for a variable or a set of variables to which a factor connects. For efficiency, we perform approximate inference using belief propagation, running it until convergence. We use minimum Bayes risk decoding, where we compute the marginals for each variable in our model and independently return the most likely setting of each variable. Marginals typically converge in 3–5 iterations of belief propagation, so we use 5 iterations in our experiments.

6.2 Evaluation

6.2.1 Experimental Setup

We perform training and evaluation on the KBP 2017 English and Chinese corpora. For English, we train models on 646 of the training documents, tune parameters on 171 training documents, and report results on the official KBP 2017 English test set. For Chinese, we train models on 438 of the training documents, tune parameters on 110 training documents, and report results on the official KBP 2017 Chinese test set.

Results of event coreference and trigger detection are obtained using version 1.8 of the official scorer provided by the KBP 2017 organizers. To evaluate event coreference performance, the scorer employs four commonly-used scoring measures, namely MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF_e (Luo, 2005) and BLANC (Recasens and Hovy, 2011), as well as the unweighted average of their F-scores (AVG-F). The scorer reports event mention detection performance in terms of Precision (P), Recall (R) and F-score, considering a mention correctly detected if it has an exact match with a gold mention in terms of boundary and event subtype.

Table 6.2. Results of event coreference and trigger detection on the KBP 2017 English and Chinese test sets.

		Event Coreference						Trigger Detection			
	English	MUC	B^3	$CEAF_e$	BLANC	AVG-F	Δ	P	R	F	Δ
1	Jiang et al. (2017a)	30.63	43.84	39.86	26.97	35.33		56.83	55.57	**56.19**	
2	Full	37.11	44.49	40.03	29.93	**37.89**		64.45	46.92	54.30	
3	− Topic	34.16	43.76	40.78	28.20	36.72	−1.17	64.39	46.67	54.11	−0.19
4	− Discourse	34.53	43.06	40.07	27.95	36.40	−1.49	62.15	47.49	53.84	−0.46
5	− Both	31.94	42.84	40.21	26.49	35.37	−2.52	63.57	45.87	53.29	−0.89
		Event Coreference						**Trigger Detection**			
	Chinese	MUC	B^3	$CEAF_e$	BLANC	AVG-F	Δ	P	R	F	Δ
6	Lu and Ng (2017c)	27.07	34.18	32.22	18.57	28.01		46.61	46.91	46.76	
7	Full	27.89	40.95	39.49	22.00	**32.58**		51.81	54.81	**53.27**	
8	− Topic	26.39	40.43	38.75	21.18	31.69	−0.89	51.81	53.28	52.53	−0.74
9	− Discourse	26.13	40.78	39.31	21.02	31.81	−0.77	51.65	54.65	53.11	−0.16
10	− Both	25.93	37.50	34.24	19.92	29.40	−3.18	56.78	44.63	49.98	−3.29

6.2.2 Results

Results on the English test set are shown in the top half of Table 6.2. Specifically, row 1 shows the results of Jiang et al. (2017a)'s resovler, which achieves the best performance in English trigger detection and event coreference systems participating in KBP 2017. Row 2 shows the results of our full model, which substantially outperforms the baseline system (row 1), yielding an improvement of 2.56 points in AVG-F for event coreference though it underperforms by 1.89 points in F-score for trigger detection. The higher recall of KBP 2017 system for trigger detection may partially due to the application of oversampling technique to increase the numbers of subtypes with few instances and using a ensemble model.

Results on the Chinese test set are shown in the bottom half of Table 6.2. Specifically, row 6 shows the results of Lu and Ng's (2017c) resolver, which is the top KBP 2017 system for Chinese and has produced the best results to date on this test set. Our full model (row 7) outperforms this baseline by 4.57 points in AVG-F for event coreference and 6.51 points in F-score for trigger detection. Despite the large improvement in AVG-F, the MUC F-score only increases by 0.82 points. Since MUC F-scores are computed solely based on coreference links, these results suggest that the improvement in AVG-F can largely be attributed to

Table 6.3. Statistics on salience-based candidate pruning.

		English		Chinese	
		Training	**Test**	**Training**	**Test**
1	Number of event mentions to be resolved	52370	9494	39758	9918
2	Number of candidate antecedents **before** pruning	371718	48750	124292	26406
3	Number of candidate antecedents **after** pruning	119416	20956	83378	20109
4	Number (%) of anaphoric event mentions	4362 (8.3%)	914 (9.6%)	1713 (4.3%)	821 (8.3%)
5	Number (%) of anaphoric event mentions whose correct antecedent are among the candidates **before** pruning	4317 (99.0%)	803 (87.8%)	1671 (97.6%)	585 (71.3%)
6	Number (%) of anaphoric event mentions whose correct antecedent are among the candidates **after** pruning	3171 (72.7%)	670 (73.3%)	1610 (94.0%)	565 (68.8%)

successful identification singleton clusters rather than successful identification of coreference links.

6.2.3 Model Ablations

To evaluate the importance of each of the two extensions in the full model, we perform ablation experiments. Rows 3–5 and rows 8–10 in Table 6.2 show the English and Chinese results obtained using models that are retrained after one or both of the extensions are removed from the full model. The changes in AVG-F as a result of the ablations are shown in the Δ columns for both tasks.

Similar conclusions can be drawn from the ablation results for both languages. First, ablating each of the two extensions causes a drop in performance for both event coreference and trigger detection. These results suggest that topic modeling and discourse pruning are both useful for the two tasks. Second, ablating both extensions causes a more abrupt drop in performance than ablating one of the extensions. This implies that each extension is providing useful information for each task that cannot be provided by the other extension. Third, when both extensions are ablated, the resulting models still outperform the baselines for both tasks. Nevertheless, we can see that for English, discourse pruning contributes more

to the performance of our full model than topic modeling, whereas the reverse is true for Chinese.

6.2.4 Analysis of Salience-Based Pruning

To gain insights into the effectiveness of discourse modeling in terms of pruning candidate antecedents, Table 6.3 shows some statistics on the candidate antecedents before and after applying pruning. Concretely, row 1 shows the total number of event mentions to be resolved in the English and Chinese training and test sets. For English, as we can see in rows 2–3, only 32.1% and 43.0% of the candidate antecedents remain in the training and test sets respectively after pruning. This can be attributed to the fact that we aggressively prune the candidate antecedents by allowing k (the number of top candidate antecedents that can survive the pruning for each event mention) to be in the range of 1 to 5 during parameter tuning.[8] Row 4 shows that among all event mentions to be resolved, only 8.3% of them are anaphoric. Row 5 shows that before pruning, the correct antecedent of almost all of the anaphoric event mentions in the training set is among the set of candidate antecedents, whereas the corresponding number on the test set is only 87.8% due to the presence of unseen event mentions. Row 6 shows that 72.7% and 73.3% of the correct antecedents on the training set and the test set survive the pruning, respectively. Similar trends can be observed for the Chinese datasets. Overall, these statistics shed light on why discourse-based pruning is beneficial: the percentage of correct antecedents that survive the pruning is far greater than the percentage of candidate antecedents that are pruned.

6.2.5 Discussion

One thing that the reader may not be able to appreciate just by looking at the performance numbers in Table 6.2 is that our two extensions are starting to attack some of the non-

[8]The best k according to the development set is 2 for English and 3 for Chinese.

Table 6.4. Examples illustrating the usefulness of topic modeling and salience-based pruning.

A barrage of US missile **{struck}**$_{m_1}$ Pakistan's North Waziristan tribal district on Tuesday, killing at least 15 militants.
President Vladimir Putin sent his condolences to U.S. President Barack Obama on Tuesday over the deadly tornado that **{struck}**$_{m_2}$ Oaklahoma City. The tornado **{struck}**$_{m_3}$ the southern suburbs of the Oklahoma state capital Monday afternoon, killing at least 51 people and injuring at least 140 others.
The French police said they were continuing to search for the man responsible for **{stabbing}**$_{m_4}$ a uniformed soldier in the neck Saturday evening. The soldier was **{stabbed}**$_{m_5}$ in the back of the neck with a box cutter or short knife as he patrolled with two colleagues through the transport station of La Défense, a business area in a suburb of Paris. The police suggested that the deed may have been inspired by the **{attack}**$_{m_6}$ on a British soldier in a London street Wednesday. A spokesman for the police union UNSA, Christophe Crépin, said there were similarities with the London **{attack}**$_{m_7}$. The case of the **{wounded}**$_{m_8}$ soldier, Pfc. Cédric Cordier, 23, is being handled by France's anti-terrorism court, officials said Sunday.

trivial aspects of event coreference that involve semantics and discourse, as opposed to those previous approaches that focus on low-level issues (e.g., string matching). For this reason, we will take a look at some of the errors addressed by our extensions below.

Let us first consider the kind of errors topic modeling allows us to address. Consider the first two sentences in Table 6.4, both of which contain the trigger candidate "struck". While "struck" triggers a "Conflict.Attack" event in the first sentence, neither of its occurrences in the second sentence corresponds to a true trigger (and therefore their subtypes should both be NONE). Without topic modeling, the model predicts all occurrences of "struck" in these sentences as belonging to Conflict.Attack (and hence misclassifies the subtypes of m_2 and m_3). The reasons are that (1) "struck" is most frequently associated with "Conflict.Attack" in the training data, and (2) since the two sentences have a similar syntactic structure and contain entities of the same type, the model fails to identify their differences. In contrast, with topic modeling, our model correctly predicts the topic of the document in which the second example appears as Contact.Meeting. Since the model manages to learn that the subtype of "struck" should be NONE when the topic is Contact.Meeting and that its subtype

should be "Conflict.Attack" when the topic is "Conflict.Attack", it correctly predicts m_2 and m_3 as having subtype NONE and, as a result, it also correctly determines that they are not coreferent. In other words, by using global information encoded by the topic model, our model can distinguish between words that have different meanings in different contexts.

Next, consider the last example in Table 6.4, which aims to give the reader an idea of the usefulness of discourse-based pruning. In this example, m_4, m_5, and m_8 refer to the event of the French soldier being stabbed and are coreferent, whereas m_6 and m_7 refer to the attack on the British solider and form another coreference cluster. Without discourse-based pruning, the model mistakenly links m_8 with m_7 because they both have subtype "Conflict.Attack". In contrast, discourse-based pruning ranks m_4 and m_5 higher than m_6 and m_7 in m_8's list of candidate antecedents, the reason being that m_4, m_5, and m_8 share the same entity (realized as "a uniformed soldier", "The soldier", and "the wounded soldier") in their contexts. Since the model retains only the top two candidate antecedents for English, m_6 and m_7 are being pruned, and the model successfully resolves m_8 to m_5.

6.3 Chapter Summary

We incorporated non-local information into a state-of-the-art joint model for event coreference resolution via topic modeling and discourse-based pruning. The resulting model not only significantly outperforms the independent models but also achieves the best results to date on the KBP 2017 English and Chinese event coreference corpora.

CHAPTER 7

IMPROVING EVENT COREFERENCE RESOLUTION BY LEARNING ARGUMENT COMPATIBILITY FROM UNLABELED DATA [1]

Arguments are the participants of an event, each having its role. For example, as shown in the example in Fig 7.1, *KMT* is the AGENT-argument and *new party chief* is the PATIENT-argument of m_1. Argument compatibility is an important linguistic condition for determining the coreferent status between two event mentions. Two arguments are incompatible if they do not correspond to the same real-world entity when they are expressed in the same level of specificity; otherwise, they are compatible. For example, a pair of TIME-arguments — *Wednesday* and *2005* — which are expressed in different level of specificity, are considered compatible. If two event mentions have incompatible arguments in some specific argument roles, they cannot be coreferent. For example, m_2 and m_6 are not coreferent since their TIME-arguments — *January 2012* and *2005* — and their PATIENT-arguments — *a new chairperson* and *Ma* — are incompatible. On the other hand, coreferent event mentions can only have compatible arguments. For example, m_3 and m_5 both have *Wednesday* as TIME-arguments. In this example, argument compatibility in the TIME argument role is a strong hint suggesting their coreference.

Despite its importance, incorporating argument compatibility into event coreference systems is challenging due to the lack of sufficient labeled data. Many existing works have relied on implementing argument extractors as upstream components and designing argument features that capture argument compatibility in event coreference resolvers. However, the error introduced in each of the steps propagates through these resolvers and hinders their performance considerably.

In light of the aforementioned challenge, we propose a framework for transferring argument (in)compatibility knowledge to the event coreference resolution system, specifically by

[1] This chapter was previously published in Huang et al. (2019).

[1]	KMT to **elect**$_{m1}$ new party chief
[2]	The ruling Chinese Kuomingtang Party (KMT) in Taiwan is scheduled to **elect**$_{m2}$ a new chairperson in January 2012, it was decided at the top-level KMT **meeting**$_{m3}$ on Wednesday.
[3]	KMT chairman Ma Ying-jeou stressed fair paly during the **election**$_{m4}$ at Wednesday's **meeting**$_{m5}$, saying guest-feting or gift-giving must be banned during the process.
[4]	Ma was twice **elected**$_{m6}$ the party's chairman, first in 2005 and then in 2009.
ev_1: m_1, m_2, m_4 ev_2: m_3, m_5 ev_3: m_6	

Figure 7.1. A document with three events described in six event mentions. Coreferent event mentions are highlighted with the same color.

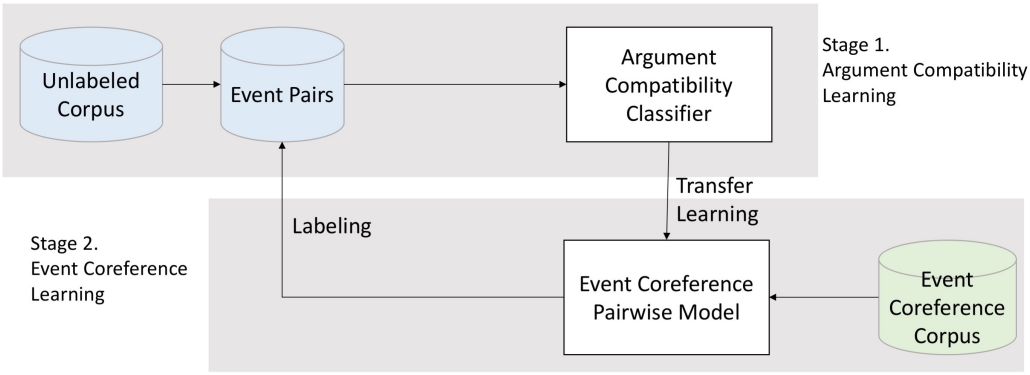

Figure 7.2. System overview.

adopting the interactive inference network (Gong et al., 2018) as our model structure. The idea is as follows. First, we train a network to determine whether the corresponding arguments of an event mention pair are compatible on automatically labeled training instances collected from a large unlabeled news corpus. Second, to transfer the knowledge of argument (in)compatibility to an event coreference resolver, we employ the network (pre)trained in the previous step as a starting point and train it to determine whether two event mentions are coreferent on manually labeled event coreference corpora. Third, we iteratively repeat the above two steps, where we use the learned coreference model to relabel the argument

Table 7.1. Examples of NER-based sample filtering. The phrases tagged as DATE are underlined, and the trigger words are boldfaced.

	event mention	DATE-compatibility with m_a
m_a	The result of the **election** last October surprised everyone.	-
m_1	He was **elected** as president in 2005.	no
m_2	The presidential **election** took place on October 20th.	yes
m_3	The opposition party won the **election**.	yes

compatibility instances, retrain the network to determine argument compatibility, and use the resulting pretrained network to learn an event coreference resolver. In essence, we mutually bootstrap the argument (in)compatibility determination task and the event coreference resolution task.

Our contributions are two-fold. First, we utilize and leverage the argument (in)compatibility knowledge acquired from a large unlabeled corpus for event coreference resolution. Second, we employ the interactive inference network as our model structure to iteratively learn argument compatibility and event coreference resolution. Initially proposed for the task of natural language inference, the interactive inference network is suitable for capturing the semantic relations between word pairs. Experimental results on the KBP coreference dataset show that this network architecture is also suitable for capturing the argument compatibility between event mentions.

7.1 Method

Our proposed transfer learning framework consists of two learning stages, the pretraining stage of an argument compatibility classifier and the fine-tuning stage of an event coreference resolver (Figure 7.2). We provide the details of both stages in sections 7.1.1 and 7.1.2, and describe the iterative strategy combining the two training stages in section 7.1.3. Details on the model structure are covered in section 7.1.4.

7.1.1 Argument Compatibility Learning

In the pretraining stage, we train the model as an argument compatibility classifier with event mentions extracted from a large unlabeled news corpus.

Task definition Given a pair of event mentions (m_a, m_b) with related triggers, predict whether their arguments are compatible or not.

Here, an event mention is represented by a trigger word and the context words within an n-word window around the trigger.

Related trigger extraction We analyze the event coreference resolution corpus and extract trigger pairs that are coreferent more than k times in the training data. We define these trigger pairs to be *related* triggers in our experiment. In this work, we set k to 10. Table 7.2 shows some examples of related triggers with high counts.

Table 7.2. Examples of related triggers.

trigger pair	count
kill - death	86
shoot - shooting	35
retire - retire	34
demonstration - protest	30

If the triggers of an event mention pair are related, their coreferent status cannot be determined by looking at the triggers alone, and this is the case in which argument compatibility affects the coreferent status most directly. Thus, we focus on the event mention pairs with related triggers in the pretraining stage of argument compatibility learning.

Compatible samples extraction From each document, we extract event mention pairs with related triggers and check whether the following conditions are satisfied:

1. DATE-compatibility (Table 7.1):

   First, we perform named entity recognition (NER) on the context words. If both event

mentions have phrases tagged as DATE in the context, these two phrases must contain at least one overlapping word. If there are multiple phrases tagged as DATE in the context, only the phrase closest to the trigger word is considered.

2. PERSON-compatibility: Similar to 1.

3. NUMBER-compatibility: Similar to 1.

4. LOCATION-compatibility: Similar to 1.

5. Apart from function words, the ratio of overlapping words in their contexts must be under 0.3 for both event mentions. We add this constraint in order to remove trivial samples of nearly identical sentences.

Conditions 1–4 are heuristic filtering rules based on NER tags, which aim to remove samples with apparent incompatibilities. Here, we consider four NER types — DATE, PERSON, NUMBER, and LOCATION — because these types of words are the most salient types of incompatibility that can be observed between event mentions. Condition 5 aims to remove event mention pairs that are "too similar". We add this condition because we do not want our model to base its decisions on the number of overlapping words between the event mentions.

We collect event mention pairs satisfying all the above conditions as our initial set of compatible samples.

Incompatible sample extraction From different documents in the corpus, we extract event mentions with related triggers and check whether the following conditions are satisfied:

1. The creation date of the two documents must be at least one month apart.

2. Apart from the trigger words and the function words, the context of the event mentions must contain at least one overlapping word.

In the unlabeled news corpus, articles describing similar news events are sometimes present. Thus, we use condition 1 to roughly assure that the event mention pairs extracted are not coreferent. Mention pairs extracted from the same document tend to contain overlapping content words, so to prevent our model to make decisions based on the existence of overlapping words, we add condition 2 as a constraint.

We collect event mention pairs satisfying all the above conditions as our initial set of incompatible samples.

Argument compatibility classifier With the initial set of compatible and incompatible samples acquired above, we train a binary classier to distinguish between samples of the two sets.

7.1.2 Event Coreference Learning

In the fine-tuning stage, we adapt the argument compatibility classifier on the labeled event coreference data to a mention-pair event coreference model.

Event Mention Detection

Before proceeding to the task of event coreference resolution, we have to identify the event mentions in the documents. We train a separate event mention detection model to identify event mentions along with their subtypes.

We model event mention detection as a multi-class classification problem. Given a candidate word along with its context, we predict the subtype of the event mention triggered by the word. If the given candidate word is not a trigger, we label it as NULL. We select the words that have appeared as a trigger at least once in the training data as candidate trigger words. We do not consider multi-word triggers in this work.

Given an input sentence, we first represent each of its comprising words by the concatenation of the word embedding and the character embedding of the word. These representation

vectors are fed into a bidirectional LSTM (biLSTM) layer to obtain the hidden representation of each word.

For each candidate word in the sentence, its hidden representation is fed into the inference layer to predict the class label. Since the class distribution is highly unbalanced, with the NULL label significantly outnumbering all the other labels, we use a weighted softmax at the inference layer to obtain the probability of each class. In this work, we set the weight to 0.1 for the NULL class label and 1 for all the other class labels.

Intuitively, candidate triggers with the same surface form in the same document tend to have the same class label. However, it is difficult to model this consistency since our model operates at the sentence level. Thus, we account for this consistency across sentences by the following post-processing step: If a candidate word is assigned the NULL label but more than half of the candidates sharing the same surface form is detected as triggers of a specific subtype, then we change the label to this given subtype. Also, we disregard event mentions with types *contact*, *movement* and *transaction* in this post-processing step, since the subtypes under these three types do not have a good consistency across different sentences in the same document.

Mention-Pair Event Coreference Model

With the argument compatibility classifier trained in the previous stage, we use the labeled event coreference corpus to fine-tune the model into an event coreference resolver. We design the event coreference resolver to be a mention-pair model (Soon et al., 2001), which takes a pair of event mentions as the input and outputs the likelihood of them being coreferent.

With the pairwise event coreference predictions, we further conduct best-first clustering (Ng and Cardie, 2002b) on the pairwise results to build the event coreference clusters of each document. Best-first clustering is an agglomerative clustering algorithm that links each event mention to the antecedent event mention with the highest coreference likelihood given the likelihood is above an empirically determined threshold.

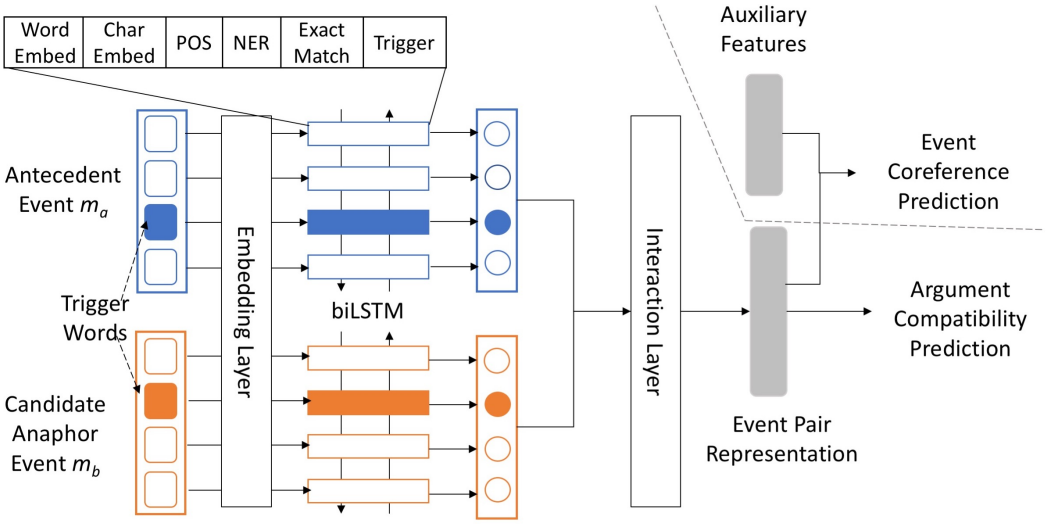

Figure 7.3. Model structure.

7.1.3 Iterative Relabeling Strategy

Previously, we collected a set of compatible event mentions from the same document with simple heuristic filtering. Despite this filtering step, the initial compatible set is noisy. Here, we introduce an iterative relabeling strategy to improve the quality of the compatible set of event mentions.

First, we calculate the coreference likelihood of the event mentions in the initial compatible set. Mention pairs with a coreference likelihood above threshold θ_M are added to the new compatible set. On the other hand, mention pairs with a coreference likelihood below θ_m are added to the initial incompatible set to form the new incompatible set. With the new compatible and incompatible sets, we can start another iteration of transfer learning to train a coreference resolver with improved quality. In this work, we set θ_M to 0.8 and θ_m to 0.2.

7.1.4 Model Structure

We adopt an interactive inference network as the model structure of our proposed method (Figure 7.3). A qualitative analysis of an interactive inference network shows that it is good

at capturing word overlaps, antonyms and paraphrases between sentence pairs (Gong et al., 2018). Thus, we believe this network is suitable for capturing the argument compatibility between two event mentions. The model consists of the following components:

Model inputs The input to the model is a pair of event mentions (m_a, m_b), with m_a being the antecedent mention of m_b:

$$\begin{aligned} m_a &= \{w_a^1, w_a^2, ..., w_a^N\} \\ m_b &= \{w_b^1, w_b^2, ..., w_b^N\} \end{aligned} \tag{7.1}$$

Each event mention is represented by a sequence of N tokens consisting of one trigger word and its context. Here, we take the context to be the words within an n-word window around the trigger. In this work, n is set to 10.

Embedding layer We represent each input token by the concatenation of the following components:

Word embedding The word representation of the given token. We use pretrained word vectors to initialize the word embedding layer.

Character embedding To identify (in)compatibilities regarding person, organization or location names, the handling of out-of-vocabulary (OOV) words is critical.

Adding character-level embeddings can alleviate the OOV problem (Yang et al., 2017). Thus, we apply a convolutional neural network over the comprising characters of each token to acquire the corresponding character embedding.

POS and NER one-hot vectors One-hot vectors of the part-of-speech (POS) tag and NER tag.

Exact match A binary feature indicating whether a given token appears in the context of both event mentions. This feature is proved useful for several NLP tasks operating on pairs of texts (Chen et al., 2017; Gong et al., 2018; Pan et al., 2018).

Trigger position We encode the position of the trigger word by adding a binary feature to indicate whether a given token is a trigger word.

Encoding layer We pass the sequence of embedding vectors into a biLSTM layer (Hochreiter and Schmidhuber, 1997), resulting in a sequence of hidden vectors of size $|h|$:

$$\begin{aligned} h_a^i &= biLSTM(emb(w_a^i), h_a^{i-1}) \\ h_b^i &= biLSTM(emb(w_b^i), h_b^{i-1}) \end{aligned} \tag{7.2}$$

where $emb(w)$ is the embedding vector of token w.

Interaction layer The interaction layer captures the relations between two event mentions based on the hidden vectors h_a and h_b. The interaction tensor I, a 3-D tensor of shape (N, N, $|h|$), is calculated by taking the pairwise multiplication of the corresponding hidden vectors:

$$I_{ij} = h_a^i \circ h_b^j \tag{7.3}$$

Finally, we apply a multi-layer convolutional neural network to extract the event pair representation vector f_{ev}.

Inference layer In the pretraining stage, we feed f_{ev} to a fully-connected inference layer to make a binary prediction of argument compatibility.

As for the fine-tuning stage, we concatenate an auxiliary feature vector f_{aux} to f_{ev} before feeding it into the inference layer. f_{aux} consists of two features, a one-hot vector that encodes the sentence distance between the two event mentions and the difference of the word embedding vectors of the two triggers.

Table 7.3. Event coreference resolution results of our proposed system, compared with the biLSTM baseline model and the current state-of-the-art system.

	MUC	B^3	$CEAF_e$	BLANC	AVG-F
biLSTM (standard)	29.49	43.15	39.91	24.15	34.18
biLSTM (transfer)	33.84	42.91	38.39	26.59	35.43
Interact (standard)	31.12	42.84	39.01	24.99	34.49
Interact (transfer)	34.28	42.93	39.95	32.12	36.24
Interact (transfer, 2^{nd} iter)	35.66	43.20	40.02	32.43	**36.75**
Interact (transfer, 3^{rd} iter)	36.05	43.07	39.69	28.06	36.72
Jiang et al. (2017)	30.63	43.84	39.86	26.97	35.33

7.2 Evaluation

7.2.1 Experimental Setup

Corpora

We use English Gigaword (Parker et al., 2009) as the unlabeled corpus for argument compatibility learning. This corpus consists of the news articles from five news sources, each annotated with its creation date.

As for event coreference resolution, we use the English portion of the KBP 2015 and 2016 datasets (Ellis et al., 2015, 2016) for training, and the KBP 2017 dataset (Getman et al., 2017) for evaluation. The KBP datasets comprise news articles and discussion forum threads. The KBP 2015, 2016, and 2017 corpora contain 648, 169, and 167 documents, respectively. Each document is annotated with event mentions of 9 types and 18 subtypes, along with the coreference clusters of these event mentions.

Implementation Details

Preprocessing We use the Stanford CoreNLP toolkit (Manning et al., 2014) to perform preprocessing on the input data.

Network structure Each word embedding is initialized with the 300-dimensional pre-trained GloVe embedding (Pennington et al., 2014). The character embedding layer is a

combination of an 8-dimensional embedding layer and three 1D convolution layers with a kernel size of 5 with 100 filters. The size of the biLSTM layer is 200. The maximum length of a word is 16 characters; shorter words are padded with zero and longer words are cropped. For the interaction layer, we use convolution layers with a kernel size of 3 in combination with max-pooling layers. The size of the inference layer is 128. Sigmoid activation is used for the inference layer, and all other layers use ReLU as the activation function.

Event mention detection model For word embeddings, we use the concatenation of a 300-dimensional pretrained GloVe embedding and the 50-dimensional embedding proposed by Turian et al. (2010). The character embedding layer is a combination of an 8-dimensional embedding layer and three 1D convolution layers with kernel sizes of 3, 4, 5 with 50 filters.

Evaluation Metrics

We follow the standard evaluation setup adopted in the official evaluation of the KBP event nugget detection and coreference task. This evaluation setup is based on four distinct scoring measures —MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF$_e$ (Luo, 2005) and BLANC (Recasens and Hovy, 2011) — and the unweighted average of their F-scores (AVG-F). We use AVG-F as the main evaluation measure when comparing system performances.

7.2.2 Results

We present the experimental results on the KBP 2017 corpus in Table 7.3. In the following, we compare the performance of methods with different network architectures and experimental settings.

Comparison of network architectures We compare the results of the interactive inference network (Interact) with the biLSTM baseline model (biLSTM).

Table 7.4. Examples of event pairs with related triggers. Trigger words are boldfaced, and words with (in)compatibility information are colored in blue.

Type	Event Mention Pair	Gold	System
Explicit	m_1: ... the building where 13 people were **killed** will be razed, and a memorial ... m_2: In that case, George Hennard **killed** 23 people at a Luby 's restaurant, ...	non-coref	non-coref
Explicit	m_1: Ten relatives of the victims arrived at the airport Sunday before **traveling** to the city of Jiangshan. m_2: On Monday , the victims' relatives **went** to the Jiangshan Municipal Funeral Parlor.	non-coref	non-coref
Implicit	m_1: ... a young woman **protester** was brutally slapped while she was demonstrating ... m_2: ... explain why a women **protester** in her 60s was beaten up by policemen ...	non-coref	coref
Implicit	m_1: She died from a brain hemorrhage on July 10, 2003, ... m_2: ... has denied killing his second wife, whom he says **died** in a car accident.	non-coref	non-coref
Implicit	m_1: Nationwide demonstrations held in France to **protest** gay marriage. m_2: ... to **protest** against the country's plan to legalize same-sex marriage.	coref	coref
General	m_1: ... Connecticut elementary school **shooting** has reignited the debate over gun control. m_2: Gun supporters hold that people, not guns, are to blame for the **shootings**.	non-coref	coref
General	m_1: Industrial accidents have injured and **killed** Foxconn workers, and the company also experienced ... m_2: ... explosion in May 2011 at Foxconn 's Chengdu factory **killed** three workers ...	non-coref	non-coref

The biLSTM baseline model does not have the interaction layer. Instead, the last hidden vectors of the biLSTM layer are concatenated and fed into the inference layer directly.

When trained solely on the event coreference corpus (standard), the model with the interactive inference network performs slightly better than the biLSTM baseline model, as shown in rows 1 and 3. However, with an additional pretraining step of argument compatibility learning (transfer), the interact inference network outperforms the biLSTM baseline model by a considerable margin, as shown in rows 2 and 4. We conclude that the interactive inference network can better capture the complex interactions between two event mentions, accounting for the difference in performance.

Effect of transfer learning Regardless of the network structure, we observe a considerable improvement in performance by pretraining the model as an argument compatibility classifier. The biLSTM baseline model achieves an improvement of 1.25 points in AVG-F by doing transfer learning, as can be seen in rows 1 and 2. As for the interactive inference network, an improvement of 1.75 points in AVG-F is achieved, as can be seen in rows 3 and 4. These results provide suggestive evidence that our proposed transfer learning framework, which utilizes a large unlabeled corpus to perform argument compatibility learning, is effective.

Effect of iterative relabeling We achieve another boost in performance by using the trained event coreference resolver to relabel the training samples for argument compatibility learning. The best result is achieved after two iterations (row 5) with an improvement of 2.26 points in AVG-F compared to the standard interactive inference network (row 3). However, we are not able to obtain further gains with more iterations of relabeling (row 6). We speculate that the difference in event coreference model predictions across different iterations is not big enough to have a perceivable impact, but additional experiments are needed to determine the reason.

Comparison with the state of the art Comparing row 5 and 7, we can see that our method outperforms the previous state-of-the-art model (Jiang et al., 2017b) by 1.42 points in AVG-F.

7.3 Discussion

In this section, we conduct a qualitative analysis of the outputs of our best-performing system (the Interact (transfer, 2^{nd} iter) system in Table 7.3) on the event coreference dataset and the unseen event mention pairs extracted from the unlabeled corpus.

7.3.1 Compatibility Classification

We focus on the samples with related triggers having either compatible or incompatible arguments (Table 7.4). These samples can be roughly classified into the following categories:

Explicit argument compatibility The existence of identical/distinct time phrases, numbers, location names or person names in the context is the most explicit form of (in)compatibility.

For these event pairs, the existence of identical/distinct phrases with the same NER type is a direct clue toward deciding their coreferent status. Making use of this nature, we perform

filtering on the set of compatible samples acquired from the unlabeled corpus in order to remove samples with explicit incompatibility.

Our model can recognize this type of (in)compatibility with a relatively high accuracy. Both examples shown in Table 7.4 are classified correctly.

Implicit argument compatibility Event pairs with implicit (in)compatible arguments require external knowledge to resolve.

We present three examples in Table 7.4. In the first example, the knowledge that a woman *in her 60s* is generally not referred to as being *young* is required to determine the incompatibility. Similarly, the knowledge that both *brain hemorrhage* and *car accident* are causes of people's death are required to classify the second example correctly.

While the performance on samples with implicit (in)compatibility is not as good as that on samples with explicit (in)com-patibility, our system is able to capture implicit (in)compatibility to some extent. We believe that this type of (in)compatibility is difficult to be captured with the argument features that are designed based on the outputs of argument extractors and entity coreference resolvers, and that the ability to resolve implicit (in)compatibility contributes largely to our system's performance improvements.

General-specific incompatibilities Event mentions describing general events pose special challenges to the task of event coreference resolution.

In Table 7.4, we present two typical examples of this category. In the first example, the second event mention does not refer to any specific shooting event in the real world, in contrast to the first event mention, which describes a specific school shooting event. Similarly for the second example, where the first event mention depicts a general event and the second event mention depicts a specific one.

General event mentions typically have few or even no arguments and modifiers, making the identification of non-coreference relations very challenging. Since we cannot rely on

Table 7.5. Case study on manually-generated event mention pairs. Trigger words are bold-faced, and the target arguments are colored in blue.

	Event Mention Pair	Type	System
1	m_1: What would have happened if Steve Jobs had never **left** Apple ...	-	-
	m_2^a: ...in the state that is today if John hadn't **left**.	Explicit	non-coref
	m_2^b: ...in the state that is today if she hadn't **left**.	Implicit	non-coref
	m_2^c: ...in the state that is today if he hadn't **left**.	Implicit	coref
2	m_1: Police arrest 6 men for **gangraping** housewife in northern India.	-	-
	m_2^a: Indian police have arrested six men for allegedly **gangraping** a 29-year-old housewife ...	Explicit	coref
	m_2^b: Indian police have arrested six men for allegedly **gangraping** a woman ...	Implicit	coref
	m_2^c: Indian police have arrested six men for allegedly **gangraping** a medical student ...	Implicit	non-coref
3	m_1: Nationwide demonstrations in France to **protest** gay marriage.	-	-
	m_2^a: ...took to the streets across the country to **protest** against the country's plan to legalize same-sex marriage.	Implicit	coref
	m_2^b: ...took to the streets across the country to **protest** against the contentious citizenship amendment bill.	Implicit	non-coref

argument compatibility, a deeper understanding of the semantics of the event mentions is needed. General event mentions account for a considerable fraction of our system's error, since they are quite pervasive in both news articles and discussion forum threads.

7.3.2 Case Study

To better understand the behavior of our system, we perform a case study on manually-generated event pairs. Specifically, for a given pair of event mentions, we first alter only one of the arguments and keep the rest of the content fixed. We then observe the behavior of the system across different variations of the altered argument (Table 7.5).

Example 1 In this example, we pick the AGENT-argument as the target and alter the AGENT-argument of the second event mention. The event pair (m_1, m_2^a) is non-coreferent due to the explicit incompatibility between *Steve Jobs* and *John*, and the system's prediction is also non-coreferent. Further, we alter the target argument to the pronoun *she* (m_2^b), resulting in an implicit incompatibility in the AGENT argument since the *Steve Jobs* is generally not considered a feminine name. As expected, the system classifies the event pair (m_1, m_2^b) as non-coreferent. Finally, when we alter the target argument to *he* (m_2^c), the system correctly classifies the resulting pair as coreferent.

Example 2 In this example, we pick the PATIENT-argument as the target and alter the PATIENT-argument of the second event mention. The system classifies the event pair (m_1, m_2^a) as coreferent, which is reasonable considering the presence of the explicit compatible arguments *housewife* and *29-year-old housewife*. Further, when we alter the target argument to *woman* (m_2^b), the system output is still coreferent. This is consistent with our prediction: the event mentions are likely to be coreferent judging only from the context of the two event mentions. However, when we alter the target argument to *medical student* (m_2^c), the event pair would become non-coreferent due to the incompatibility between *medical student* and *housewife*. The system classifies the event pair correctly.

Example 3 In this example, we pick the REASON-argument as the target and alter the REASON-argument of the second event mention. The event pair (m_1, m_2^a) has a pair of implicit compatible arguments in the REASON-argument role and is likely to be coreferent. In contrast, altering the target argument to *contentious citizenship amendment bill* (m_2^b) would yield an pair of implicit incompatible arguments, and the resulting event pair would become non-coreferent. Our system classifies both event pairs correctly.

7.4 Chapter Summary

We proposed an iterative transfer learning framework for event coreference resolution. Our method exploited a large unlabeled corpus to learn a wide range of (in)compatibilities between arguments, which contributes to the improvement in performance on the event coreference resolution task. Our model outperforms the previous state-of-the-art system. In addition, a qualitative analysis of the system output confirmed the ability of our system to capture (in)compatibilities between two event mentions.

CHAPTER 8

CONSTRAINED MULTI-TASK LEARNING FOR EVENT COREFERENCE RESOLUTION [1]

As we discussed in the previous sections, one of the most common approaches to event coreference resolution is *pipelined* approaches, where a *trigger detection* component, which identifies triggers and assigns event subtypes to them, is followed by an *event coreference* component, which clusters coreferent event mentions. It should therefore not be surprising that errors propagate from the trigger detection component to the event coreference component. To avoid further aggravating this error propagation problem, information from other IE components such as entity coreference and arguments are typically employed as features for training the event coreference model (Chen et al., 2009; McConky et al., 2012; Cybulska and Vossen, 2013; Araki et al., 2014; Liu et al., 2014a; Peng et al., 2016; Krause et al., 2016; Choubey and Huang, 2017). Oftentimes, these features provide limited improvements to event coreference models as they are too noisy to be useful.

Though less popular than pipelined approaches, *bootstrapping* approaches have been used for event coreference resolution, where an event coreference model is bootstrapped with models trained for one or more related IE tasks. For instance, Lee et al. (2012) incrementally builds clusters of event and entity mentions by iteratively bootstrapping event coreference output using entity coreference output and vice versa. While in pipelined approaches only upstream tasks can influence downstream tasks, in bootstrapping approaches different tasks can influence each other. Nevertheless, errors made in earlier iterations of the bootstrapping process cannot be undone in later iterations.

Joint learning approaches have recently emerged as a promising approach to event coreference resolution owing to its ability to address error propagation. The key advantage of

[1] This chapter was previously published in Lu and Ng (2021).

these models is that the tasks involved can benefit from each other during training. However, since a jointly learned model involves multiple tasks, it is typically complex. In fact, it is by no means easy to scale such a model to a large number of tasks because of the high computational complexity involved in the learning process.

Joint inference approaches have also been applied to event coreference resolution. Since the models are trained independently, they cannot benefit from each other and could be noisy. Worse still, performing joint inference using hard constraints over (very) noisy outputs could do more harm than good. For instance, if two event mentions are correctly classified as coreferent but one of their subtypes is misclassified, then enforcing the aforementioned constraint might cause the joint inference procedure to incorrectly infer that the two are not coreferent. This explains why joint inference approaches have become less popular than joint learning approaches in recent years.

In light of the above discussion, we propose a model that jointly learns *six* tasks: trigger detection, event coreference, entity coreference, anaphoricity determination, argument extraction, and realis detection. As noted above, joint learning typically presents a serious computational challenge, and training a complex joint model involving six tasks would not have been possible without the advent of neural NLP era.

While multi-task learning in a neural network typically allows the different tasks involved to benefit from each other via learning shared representations, we hypothesize that the model would benefit additional guidance given that the learning task (which involves six tasks) is so complex. Consequently, we propose to guide the learning process by exploiting cross-task consistency constraints. As mentioned above, such consistency constraints are typically employed in joint inference and rarely in joint learning. Moreover, unlike in joint inference where such constraints are typically implemented as hard constraints, we provide flexibility by implementing them as soft constraints. Specifically, we design penalty functions for penalizing outputs that violate a constraint, where the degree of penalty depends on the extent of the violation.

Another contribution of our work involves proposing the idea of a *unified* coreference model. So far, entity and event coreference have always been viewed as two separate tasks, where links between entity mentions are distinguished from links between event mentions. However, their similarity led us to hypothesize that they could be viewed as a single task, where coreference links are established between a set of mentions without distinguishing between entity and event mentions.

8.1 Model

We design a *span-based* neural model for event coreference resolution owing to their ability to effectively learn representations of text *spans* (as opposed to words). While span-based models have been successfully applied to a variety of entity-based IE tasks such as entity coreference (Lee et al., 2017; Joshi et al., 2020) and relation extraction (Luan et al., 2019), they have not been applied to event coreference.

More formally, our model takes as input a document D represented as a sequence of word tokens, from which we extract all possible intra-sentence spans of up to length L. In event coreference resolution, each such span corresponds to a candidate trigger. Our model simultaneously learns six tasks, which we define below.

The *trigger detection* task aims to assign each span i a subtype label y_i. Each y_i takes a value in a subtype inventory or NONE, which indicates that i is not a trigger. The model predicts i's subtype to be $y_i^* = \arg\max_{y_t} s_t(i, y_t)$, where s_t is a scoring function suggesting i's likelihood of having y_i as its subtype.

The *event coreference resolution* task aims to assign span i an antecedent y_c, where $y_c \in \{1, \ldots, i-1, \epsilon\}$. In other words, the value of each y_c is the id of its antecedent, which can be one of the preceding spans or a dummy antecedent ϵ (if the event mention underlying i starts a new cluster). We define the following scoring function:

$$s_c(i,j) = \begin{cases} 0 & j = \epsilon \\ s_m(i) + s_m(j) + s_p(i,j) & j \neq \epsilon \end{cases} \tag{8.1}$$

where $s_m(i)$ is the score suggesting span i's likelihood of being a trigger and $s_p(i,j)$ is a pairwise coreference score computed over span i and a preceding span j. The model predicts the antecedent of i to be $y_c^* = \arg\max_{j \in \mathcal{Y}(i)} s_c(i,j)$, where $\mathcal{Y}(i)$ is the set containing all candidate antecedents.

The *entity coreference resolution* task involves identifying entity mentions that refer to the same real-world entity. Intuitively, entity coreference is useful for event coreference: two event mentions are not likely to be coreferent if there exists an argument role (e.g., ATTACKER) for which the corresponding arguments in the two event mentions are not entity-coreferent. In our model, it is defined in the same way as the event coreference resolution task except that it operates on the spans identified by the *entity mention detection component* rather than the trigger detection component. The entity mention detection task is defined in the same way as the trigger detection task except that it aims to assign each span i an entity type label.

The *anaphoricity determination* task aims to assign each span i an anaphoricity label y_a, where y_a can be ANAPHORIC, which indicates that the mention having span i is coreferent with a preceding mention, or NON-ANAPHORIC. The model $s_a(i)$ predicts the mention having span i as anaphoric if and only if $s_a(i) \geq 0$. To train this model, we set the target value to 1 for anaphoric mentions and -1 for non-anaphoric mentions. Anaphoricity is useful for coreference: it prevents non-anaphoric mentions from being (erroneously) resolved.

The *realis detection* task aims to assign each span i a realis label y_r, where $y_r \in \{$ACTUAL, GENERIC, OTHER, ENTITY, and NONE$\}$, indicating whether an event actually happened or will happen in the future or whether it is a generic event. ENTITY is a label that is exclusively reserved for spans that correspond to entity mentions. ACTUAL, GENERIC, and OTHER are labels used for event mention spans. NONE indicates that i does not correspond

to a mention. The model predicts the realis type of i to be $y_r^* = \arg\max_{y_r} s_r(i, y_r)$, where s_r is a scoring function suggesting i's likelihood of having realis type y_r. Realis detection is useful for event coreference: two event mentions cannot be coreferent if their realis labels are different.

The *argument extraction* task aims to assign an argument role label y_o to an argument candidate k given a span i. y_o can be a role taken from an argument role inventory or NONE, which indicates that the token is not an argument of i. For every event mention candidate span i, we consider an entity mention candidate span k an argument candidate of i if and only if it appears within the same sentence as i. For each argument candidate k of i, the model predicts its role to be $y_o^* = \arg\max_{y_o} s_o(i, k, y_o)$, where s_o is a scoring function suggesting token k's likelihood of being an argument of i having role y_o. As mentioned above, event arguments, when combined with entity coreference information, would be useful for event coreference.

8.1.1 Model Structure

The model structure, which is shown in Figure 8.1, is described in detail below.

Span Representation Layer We adapt the independent version of Joshi et al.'s (2019) state-of-the-art entity coreference resolver to event coreference resolution. Specifically, we divide an input document into non-overlapping regions, each of which has size L_d. The word sequence in each region serves as an input training sequence. We then pass the sequence into a pretrained transformer encoder to encode tokens and their contexts used in SpanBERT-large (Joshi et al., 2020). Finally, we set $\mathbf{g}_i$, the representation of span i, to $[\mathbf{h}_{start(i)}; \mathbf{h}_{end(i)}; \mathbf{h}_{head(i)}; \mathbf{f}_i]$, where $\mathbf{h}_{start(i)}$ and $\mathbf{h}_{end(i)}$ are the hidden vectors of the start and end tokens of the span, $\mathbf{h}_{head(i)}$ is an attention-based head vector and $\mathbf{f}_i$ is a span width feature embedding. To maintain computational tractability, we first compute a score s_m for each span i:

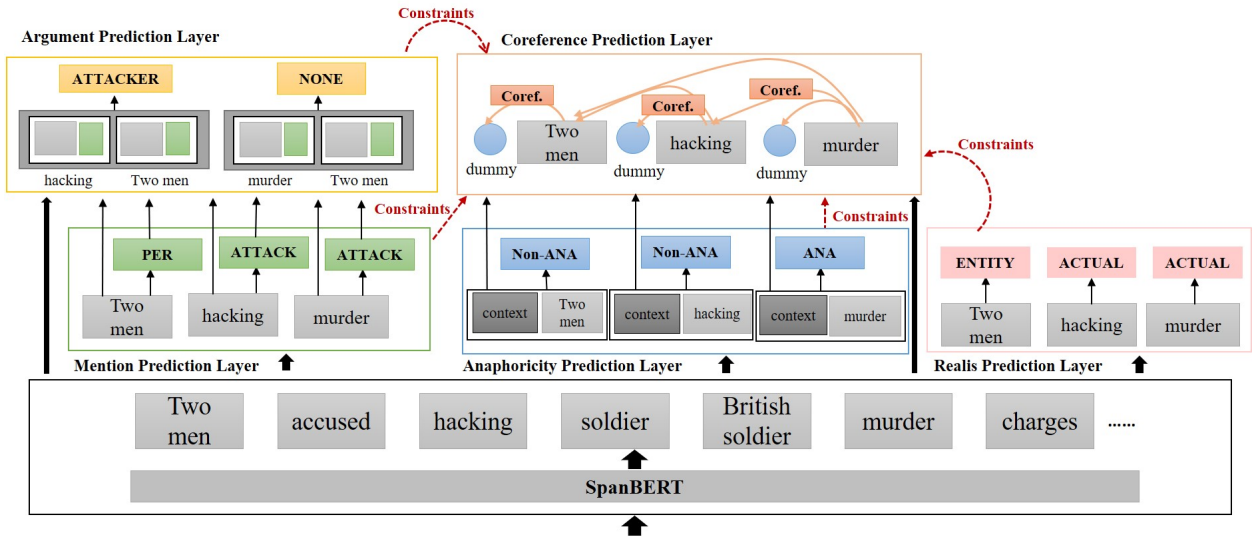

Figure 8.1. Model structure.

$$s_m(i) = \text{FFNN}_m(\mathbf{g}_i) \tag{8.2}$$

where FFNN is a standard feedforward neural network. Then we retain only the top $N\%$ of the spans for further processing.

Trigger Prediction Layer For each span i that survives the filtering, we pass its representation $\mathbf{g}_i$ to a FFNN, which outputs a vector $\mathbf{ot}_i$ of dimension T, where T is the number of possible event subtypes (including NONE). $\mathbf{ot}_i(y)$, the yth element of $\mathbf{ot}_i$, is a score indicating i's likelihood of belonging to event subtype y. Specifically:

$$\mathbf{ot}_i = \text{FFNN}_t(\mathbf{g}_i) \tag{8.3}$$

$$s_t(i, y) = \mathbf{ot}_i(y) \tag{8.4}$$

Anaphoricity Prediction Layer We predict the anaphoricity value of each top span i as follows. Since the anaphoricity of a mention is dependent on its preceding context, we first concatenate the average of the representations of the 25 tokens immediately preceding i (to approximate i's preceding context) with the span representation $\mathbf{g}_i$. We then pass the resulting vector, $\mathbf{cx}_i$, to a FFNN, which outputs an anaphoricity value. Specifically:

$$s_a(i) = \text{FFNN}_a(\mathbf{cx}_i) \tag{8.5}$$

Realis Prediction Layer To predict the realis value of each top span i, we pass its representation $\mathbf{g}_i$ to a FFNN, which outputs a vector $\mathbf{or}_i$ of length 5. $\mathbf{or}_i(y)$, the yth element of $\mathbf{or}_i$, is a score indicating i's likelihood of having realis type y:

$$\mathbf{or}_i = \text{FFNN}_r(\mathbf{g}_i) \tag{8.6}$$

$$s_r(i, y) = \mathbf{or}_i(y) \tag{8.7}$$

Coreference Prediction Layer To predict event coreference links, we define the pairwise score between span i and span j as follows:

$$s_p(i, j) = \text{FFNN}_c([\mathbf{g}_i; \mathbf{g}_j; \mathbf{g}_i \circ \mathbf{g}_j, \mathbf{u}_{ij}]) \tag{8.8}$$

where $\circ$ denotes element-wise multiplication, $\mathbf{g}_i \circ \mathbf{g}_j$ encodes the similarity between span i and span j, and $\mathbf{u}_{ij}$ is a feature embedding encoding the distance between two spans. We can then compute the full coreference score defined in equation 8.1 using equations 8.2 and 8.8.

To improve running time, we follow Lee et al. (2018) and use their antecedent pruning method, coarse-to-fine pruning, to reduce the number of candidate antecedents for each anaphor.

Incorporating Entity Coreference The most straightforward way to incorporate entity coreference information into our model would be to have (1) an entity mention detection model that is architecturally identical to the trigger detection model except that it assigns entity type (rather than subtype) labels to each span, and (2) an entity coreference model that is architecturally identical to the event coreference model described above except that it identifies antecedents for spans provided by the entity mention detection (rather than trigger detection) component. While this would allow entity coreference to interact with event coreference and other tasks via the shared span representation layer, the two coreference tasks would otherwise be learned independently of each other.

Towards the goal of building a *unified* model of coreference, we propose a novel idea: we seek to learn entity and event coreference *simultaneously* by viewing them as a *single* coreference task. In other words, from a learning perspective, there is only one task to be learned, which is coreference resolution over a set of mentions. To do so, we extend aforementioned the Span Representation Layer, the Trigger Prediction Layer, and the Coreference Layer as follows. First, the Span Representation Layer will identify spans corresponding to *mentions* that are composed of both entity mentions and event mentions even though the model doesn't know (and doesn't need to know) which ones are entity mentions and which ones are event mentions. Second, the Trigger Prediction Layer will assign each mention span a semantic type, which is taken from a type inventory consisting of both entity types and event subtypes (or NONE if the span is not a mention). In other words, the Trigger Detection Layer, which is essentially extended to a Mention Detection Layer, now extracts both entity and event mention spans. Third, the Coreference Layer computes coreference chains based on the predicted mention spans and their semantic types. Since all the learner sees are mentions, it doesn't know (and doesn't need to know) which coreference chains it computes are entity-based and which ones are event-based. Similarly, it doesn't know (and doesn't need to know) which types in the type inventory are entity types and which ones are

event subtypes. A key advantage of this unified model of coreference is that it allows entity and event coreference to be tightly coupled via parameter sharing.

When we apply this model to a test document, we need to distinguish which coreference relations it identifies are entity-based and which ones are event-based. This can be done easily based on the semantic type associated with the mentions underlying the extracted coreference relation under consideration. If the semantic type is an entity type, the corresponding coreference relation is regarded as an entity coreference relation; otherwise, it is regarded as an event coreference relation.

Argument Prediction Layer To predict arguments and their roles, we pair each top span i and each candidate argument k to form an input vector $\mathbf{va}_{ik} = [\mathbf{g}_i; \mathbf{t}_i; \mathbf{g}_k; \mathbf{t}_k]$, where $\mathbf{g}_i$ is the span representation of i, $\mathbf{t}_i$ is the one-hot subtype vector of i, $\mathbf{g}_k$ is the span representation of argument candidate k, $\mathbf{t}_k$ is the one-hot subtype vector of k respectively. During training, we use the gold subtype label to derive the subtype vector. During inference, we derive the subtype vector from the output of the mention detection layer. We feed the resulting vector into a FFNN, which outputs a vector $\mathbf{oa}_{ik}$ of dimension 21. $\mathbf{oa}_{ik}(y)$, the yth element of $\mathbf{oa}_{ik}$, is a score indicating k's likelihood of being an argument of i with role y:

$$\mathbf{oa}_{ik} = \text{FFNN}_{oa}(\mathbf{va}_{ik}) \tag{8.9}$$

$$s_o(i, k, y) = \mathbf{oa}_{ik}(y) \tag{8.10}$$

Incorporating Consistency Constraints As noted before, we propose to guide the learning process by incorporating commonsense knowledge that encodes *cross-task consistency constraints* on event coreference and the auxiliary tasks. We begin by incorporating two consistency constraints on the outputs of event coreference and trigger detection: **C1**:

If two spans are coreferent, they should have the same trigger subtype. **C2**: If a span has an antecedent that is not the dummy antecedent, its subtype shouldn't be NONE.

We incorporate each constraint into the model via a scoring function that computes how much two spans i, an anaphor, and j, a candidate antecedent of i, should be penalized if a constraint is violated. For constraint **C1**, we define a cost function, c_1, which is computed as follows:

$$c_1(i,j) = \min(|s_t(i,y_i) - s_t(i,y_j)|, |s_t(j,y_j) - s_t(j,y_i)| \tag{8.11}$$

$y_i = \arg\max_{y_t} s_t(i,y_t)$ and $y_j = \arg\max_{y_t} s_t(j,y_t)$. Intuitively, c_1 provides an estimate of the least amount of adjustment needed to make i's subtype the same as j's or the other way round. In particular, c_1 returns 0 (i.e., no penalty) if the two spans already have the same subtype.

Similarly, for constraint **C2**, we define a cost function c_2, which is computed as follows:

$$c_2(i,j) = \begin{cases} 0 & \arg\max_{y \in \mathcal{Y}} s_t(i,y) \neq \text{None} \\ s_t(i,\text{None}) - \max_{y \in \mathcal{Y} \setminus \{\text{None}\}} s_t(i,y) & \text{otherwise} \end{cases} \tag{8.12}$$

where $\mathcal{Y}$ is the set of possible subtypes. Intuitively, c_2 estimates the minimum amount that needs to be adjusted so that anaphor j's subtype is not NONE.

Finally, we incorporate c_1 and c_2 into the model as penalty terms in s_c (Equation (8.1)). Specifically, we redefine s_c as follows:

$$s_c(i,j) = \begin{cases} 0 & j = \epsilon \\ s_m(i) + s_m(j) + s_p(i,j) - [\beta_1 c_1(i,j) + \beta_2 c_2(i,j)] & j \neq \epsilon \end{cases} \tag{8.13}$$

where β_1 and β_2 are positive constants that control the *hardness* of the constraints. The smaller a β_i is, the softer the corresponding constraint is. Intuitively, if a constraint is violated, $s_c(i,j)$ will be lowered by one or more of the penalty terms, and j will less likely be selected as the antecedent of i.

In addition, we enforce the following consistency constraints. Like **C1** and **C2**, each of them will be accompanied by a cost function that will eventually be incorporated into s_c as a penalty term.

Event coreference and anaphoricity. **C3**: If a span's antecedent is not the dummy antecedent, its anaphoricity should be ANAPHORIC. **C4**: If a span has a dummy antecedent, its anaphoricity should be NON-ANAPHORIC.

Event coreference and realis detection. **C5**: If two spans are coreferent, they should have the same realis value. **C6**: If a span's antecedent is not the dummy antecedent, its realis should not be NONE.

Event coreference and argument extraction. **C7**: If two spans are coreferent and have arguments with the same role, the arguments should be coreferent.

8.1.2 Training

The loss function we use, $L(\Theta)$, is composed of the losses of the six tasks, and is defined as follows:

$$L(\Theta) = \sum_{i=1}^{d} (\lambda_c L_c + \lambda_t L_t + \lambda_a L_a + \lambda_r L_r + \lambda_o L_o) \tag{8.14}$$

where hyperparameters (i.e., the λ's) determine the trade-off between the task losses. The model is trained to minimize $L(\Theta)$, whereas the hyperparameters are tuned using grid search to maximize AVG-F (the standard event coreference evaluation metric; see the next section) on development data.

Task Losses We employ a max-margin loss for each of the six tasks.

Defining the coreference loss is slightly tricky since the coreference annotations for each document are provided in the form of clusters. We adopt the coreference loss function previously defined by Wiseman et al. (2015b) for entity coreference resolution. Specifically, let $\text{GOLD}_c(i)$ denote the set of spans preceding span i that are coreferent with i, and y_c^l be $\arg\max_{y \in \text{GOLD}_c(i)} s_c(i, y)$. In other words, y_c^l is the highest scoring (latent) antecedent of i

according to s_c among all the antecedents of i. The loss function for coreference is defined as:

$$L_c(\Theta) = \sum_{i=1}^{n} \max_{j \in \mathcal{Y}(i)} (\Delta_c(i,j)(1 + s_c(i,j) - s_c(i, y_c^l)) \tag{8.15}$$

where $\Delta_c(i,j)$ is a mistake-specific cost function that returns the cost associated with a particular type of error (Durrett and Klein, 2013).[2] Intuitively, the loss function penalizes a span i if the predicted antecedent j has a higher score than the correct latent antecedent y_c^l.

We similarly define the loss for trigger detection:

$$L_t(\Theta) = \sum_{i=1}^{n} \sum_{\hat{l} \neq y_t} \max(0, \Delta_t(i, \hat{l})(1 + s_t(i, \hat{l}) - s_t(i, y_t))) \tag{8.16}$$

where $\Delta_t(i, \hat{l})$ is a mistake-specific cost function that returns the cost associated with a particular type of error.[1] Intuitively, the loss function penalizes each span for which each of the wrong subtypes $\hat{l}$ has a higher score than the correct subtype y_t according to s_t.

The task losses for anaphoricity determination, realis detection, and argument extraction are all max-margin losses that are defined similarly as the one used for trigger detection.

8.2 Evaluation

8.2.1 Experimental Setup

Corpus

We perform training and evaluation on the English corpora used in the TAC KBP 2017 Event Nugget Detection and Coreference task. There are no official training sets: the task organizers simply made available a number of event coreference-annotated corpora for training.

[2]Space limitations preclude a description of these error types. See Durrett and Klein (2013) for details.

We use LDC2015E29, E68, E73, and E94, and LDC2016E64 as our training set, which contain 817 documents with 22894 event mentions distributed over 13146 coreference chains[3]. Among these 817 documents, we reserve 82 documents for parameter tuning and use the remaining documents for model training. We report results on the official test set, which consists of 167 documents with 4375 event mentions distributed over 2963 coreference chains.

Evaluation Metrics

Results of event coreference, trigger detection and realis detection are obtained using version 1.8 of the official scorer provided by the KBP 2017 organizers. For event coreference, the scorer employs four scoring measures, MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF_e (Luo, 2005) and BLANC (Recasens and Hovy, 2011), as well as the unweighted average of their F-scores (AVG-F). Results of trigger detection and realis detection are both expressed in terms of Precision (P), Recall (R) and F-score. The scorer considers (1) a trigger correctly detected if it has an exact match with a gold trigger in terms of boundary, event type, and event subtype, and (2) a realis label correctly classified if it has an exact match with a gold trigger in terms of boundary and realis value.

Additionally, we express results of both argument detection and anaphoricity determination in terms of Precision (P), Recall (R) and F-score. We consider an event argument correctly detected if it has an exact match with a gold trigger-argument pair in terms of trigger boundary, trigger subtype, argument head and argument role. We consider an anaphoric mention correct if it has an exact match with the boundary of a gold anaphoric mention.

Finally, we report entity coreference results in terms of CoNLL score, which is the unweighted average of MUC, B^3, and CEAF_e.

[3]LDC2015E73 and E94 don't have annotations for entity detection, entity coreference resolution and argument detection. We set the loss of those three tasks to 0 during training.

	Event Coreference					Trigger			Anaphoricity			Realis			Argument			Entity Coref.
	MUC	B^3	CEA	BLA	AVG	P	R	F	P	R	F	P	R	F	P	R	F	CoNLL
Jiang et al. (2017a)	30.6	43.8	39.9	27.0	35.3	56.8	55.6	56.2	–	–	–	48.0	46.9	47.4	–	–	–	–
Transfer Learning Model	35.7	43.2	40.0	32.4	36.8	56.8	46.4	51.1	–	–	–	–	–	–	–	–	–	–
Joint Learning model	37.1	44.5	40.0	29.9	37.9	64.5	46.9	54.3	–	–	–	–	–	–	–	–	–	–
Knowledge-lean	37.6	52.3	51.7	33.6	43.8	71.5	55.3	62.4	–	–	–	–	–	–	–	–	–	–
Pipeline	38.6	53.0	53.0	35.0	44.9	73.9	56.1	63.8	43.0	44.5	43.8	70.0	53.1	60.3	36.9	29.9	33.0	72.6
Full Joint	45.2	54.7	53.8	38.2	48.0	71.6	58.7	64.5	50.4	45.3	47.7	63.7	52.0	57.3	32.4	24.5	27.9	68.7

Table 8.1. Results of different resolvers on event coreference and related tasks. Results in rows 1-3 are copied verbatim from the original papers; – indicates the corresponding result is not available.

Implementation Details

We use the SpanBERT-large model in the span representation layer.[4] For each document, we split it into segments of length 512. We generate all spans of length up to 10. Each FFNN has 1 hidden layer of size 2000. The size of the width feature embedding is 20. For span pruning, we keep the top 50% of the spans. For candidate antecedent pruning, we keep the top 15 antecedents.

For training, we use document sized mini-batches. We apply a dropout rate of 0.3. Following Joshi et al. (2019), we use different learning rates for training the task parameters and the SpanBERT parameters. Specifically, the task learning rate is 1×10^{-5} and is decayed linearly, whereas the learning rate for SpanBERT is 2×10^{-4} and is decayed linearly. The hyperparameters in the loss function are 1, 1, 0.05, 0.5, 0.05 for $\lambda_c, \lambda_t, \lambda_a, \lambda_r, \lambda_o$.

8.2.2 Results and Discussion

Results are shown in Table 8.1. To gauge the performance of our model, we employ five baselines. Row 1 shows the result of our first baseline, Jiang et al.'s 2017a resolver, which is the best system participating in KBP 2017. Row 2 shows the performance of our second baseline, the transfer learning resolver in Chapter 7. Both resolvers perform trigger detection

[4]https://github.com/facebookresearch/SpanBERT

and event coreference in a pipeline fashion. The embeddings being used in these resolvers are uncontextualized. Row 3 shows our third baseline, joint learning resolver in Chapter 6.

Row 4 shows our fourth baseline, which is our model except that (1) three prediction layers (argument, realis, and anaphoricity) are removed, and (2) the remaining layers are trained to identify event mentions only (i.e., without entity mentions). This baseline mimics typical knowledge-lean approaches to event coreference resolution, which perform only trigger detection and event coreference, but is the first knowledge-lean event coreference approach implemented in a span-based framework. As we can see, this baseline performs considerably better than the joint learning resolver by 5.9 points in AVG-F for event coreference. A closer inspection of the other coreference evaluation metrics reveals that in comparison to the joint learning model, the B^3, CEAF_e and BLANC increased substantially while the MUC scores barely changed. Since MUC only rewards successful identification of coreference links, the fact that the MUC score was more or less unchanged implies that the improvement did not arise from link identification; rather, the fact that the B^3, CEAF_e and BLANC scores improved suggests that the improvement came from successful identification of singleton clusters. This is further supported by the improvement in trigger detection: the baseline's trigger detection module achieves an F-score of 62.4, outperforming the joint learning model's trigger detection module by 8.1 points in F-score. This huge improvement should not be surprising, as SpanBERT is designed to extract text spans. Overall, despite the 6-point improvement in event coreference AVG-F score, we cannot say that the successes of span-based models on entity coreference can be extended to event coreference as it largely fails to establish event coreference links.

Row 5 shows the result of our fifth baseline, which is a pipelined version of our model, designed to gauge the benefits of our joint model. Here, we first train a trigger detector, which is the same as the Mention Prediction Layer of our model trained to assign event subtypes to top spans. The resulting triggers are used to train an anaphoricity model (same

as our model's Anaphoricity Prediction Layer) and a realis detection model (same as our model's Realis Prediction Layer). Next, we train an entity coreference model, which is the same as our third baseline except that it is trained to operate on entity rather than event mention spans. Then, we train an argument extraction model (same as our model's Argument Prediction Layer) using the extracted entity mentions as candidate arguments for the triggers identifed by the trigger detection model. Finally, the outputs of all these models are used to enforce the seven constraints in our model as hard constraints: any candidate antecedent of an anaphor that violates any of the constraints is filtered prior to event coreference resolution. Overall, the performances of this baseline outperforms the fourth baseline by 0.6 points in AVG-F for event coreference and 1.4 points in F-score for trigger detection.

Row 6 shows the result of our full model, which outperforms the Pipeline model by 3.1 points in AVG-F for event coreference and establishes new state-of-the-art results. Encouragingly, the gains in AVG-F are accompanied by improvements w.r.t. all four coreference scoring metrics. In particular, the MUC score improves considerably by 6.6 points, which means that the full model has successfully identified event coreference links. In addition, we see a 0.7 point improvement in trigger detection over Pipeline, and a 12.9 points improvement in realis detection in comparison to Jiang et al. For bookkeeping purposes, we report the scores for each of the components of our model. Overall, the fact that our joint model outperform Pipeline suggests the benefits of joint modeling.

8.2.3 Model Ablations

To evaluate the contribution of different components in our model, we report in Table 8.2 the results of ablation in which we remove one component each time from the full model and retrain it.

Consistency constraints. Ablating the consistency constraints means removing all the penalty terms from s_c. The ablated system resembles what one would usually see in a multi-task learning setup, where the different tasks involved has a shared representation. As we can see from row 2, the performance of event coreference drops by 1.0 points in AVG-F, suggesting the usefulness of employing consistency constraints in a multi-task setup. The removal of consistency constraints also adversely affected entity coreference. This suggests that constraints **C7** and perhaps **C1** and **C2** are important.

Entity coreference. Next, we ablate the entity coreference component. Note that the ablation of entity coreference also causes the removal of the argument detection component since the latter relies on the outputs of entity coreference. We see from row 3 that event coreference performance drops precipitously by 2.7 points. These results suggest that entity coreference has a considerable positive impact on event coreference.

The next question is: will coreference performance go up or down if we treat entity and event coreference as two separate tasks that are learned in a typical multi-task setup? As we can see from row 4, the performance of event coreference and entity coreference drop by 0.8 points and 3 points, respectively. These results suggest that our viewing the two tasks as a single task is beneficial.

Anaphoricity determination. Next, we ablate the anaphoricity component, which involves removing both its task loss and consistency constraints. Moreover, the score s_a is set to 0. From row 5, we can see that event coreference performance drops by 0.5 point, and anaphoricity determination performance drops 0.8 points.

Realis detection. When we ablate realis detection, both the task loss and the corresponding consistency constraints are removed. The performance on event coreference and anaphoricity drops significantly, by 1.4 points and 1.0 points respectively, suggesting the usefulness of realis detection for both event coreference and anaphoricity detection.

		Event Coref.	Tri.	Ana.	Rea.	Arg.	Entity Coref.
		AVG	F	F	F	F	CoNLL
1	Full Model	48.0	64.5	47.7	57.3	27.9	68.7
2	− constraints	47.0	64.5	47.6	57.9	27.9	68.5
3	− entity coref.	45.3	63.5	45.0	58.2	–	–
4	sep. entity coref.	47.2	65.1	47.8	56.3	26.0	65.7
5	− anaphoricity	47.5	64.9	46.9	58.1	28.4	69.3
6	− realis	46.6	64.8	46.7	–	29.6	69.3
7	− argument	47.4	64.3	48.6	58.5	–	66.7

Table 8.2. Ablation results of the full model.

Argument detection. Finally, when the argument component is ablated, event coreference performance drops by 0.6 points in AVG-F. These results illustrate the importance of argument detection for event coreference.

Overall, these results suggest that each component contributes significantly to event coreference.

8.3 Chapter Summary

We proposed a multi-task neural model for event coreference resolution that (1) jointly learns six tasks, (2) uses consistency constraints to guide learning, and (3) views entity and event coreference as a single task. Our model achieves state-of-the-art results on the KBP 2017 English dataset.

CHAPTER 9

CONCLUSION

In this dissertation, we investigated knowledge-rich approaches in which we derive potentially useful knowledge for event coreference resolution from a variety of sources, including models that are trained on tasks that we believe are closely related to event coreference, statistical and linguistic features that are directly relevant to the prediction of event coreference links, as well as constraints that encode commonsense knowledge of when two event mentions should or should not be coreferent.

We first proposed a multi-pass sieve approach to event coreference resolution in Chapter 3, which resolves easy coreference links first and then different coreference links. We then proposed several joint models. We proposed a joint inference based event coreference resolver using Markov Logic Networks (MLNs) in Chapter 4. The model encodes rich NLP features implicitly by augmenting the MLN distribution with low dimensional unit clauses. We then proposed a joint learning model of event coreference resolution, trigger detection, and event anaphoricity determination in Chapter 5. The model encodes features for capturing cross-task interactions. Furthermore, we proposed two extensions to improve the joint learning model using the *non-local* information provided by a supervised topic model and salient discourse entities in Chapter 6. To leverage argument information, we proposed a transfer learning framework for event coreference resolution in Chapter 7. The model utilizes a large amount of unlabeled data to learn the argument compatibility between two event mentions and transfers argument (in)compatibility knowledge to the event coreference resolution system. Finally, we proposed a neural model of event coreference resolution that involves simultaneously learning six tasks related to event coreference in a multi-task learning framework in Chapter 8, and guide the model learning process by incorporating commonsense knowledge into the model that encodes cross-task consistency constraints on event coreference.

REFERENCES

Ahn, D. (2006). The stages of event extraction. In *ARTE*, pp. 1–8.

Allan, J., J. Carbonell, G. Doddington, J. Yamron, Y. Yang, and U. Amherst (1998). Topic Detection and Tracking Pilot Study Final Report. In *Proceedings of the Broadcast News Understanding and Transcription Workshop*, pp. 194–218.

Araki, J., Z. Liu, E. Hovy, and T. Mitamura (2014). Detecting Subevent Structure for Event Coreference Resolution. In *LREC*.

Araki, J. and T. Mitamura (2015). Joint Event Trigger Identification and Event Coreference Resolution with Structured Perceptron. In *EMNLP*, pp. 2074–2080.

Bagga, A. and B. Baldwin (1998). Algorithms for scoring coreference chains. In *Proceedings of the 1st Edition of the Language, Resources and Evaluation Conference*, pp. 563–566.

Bejan, C. and S. Harabagiu (2014). Unsupervised Event Coreference Resolution. *Computational Linguistics*.

Chambers, N., T. Cassidy, B. McDowell, and S. Bethard (2014). Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics 2*, 273–284.

Chang, C.-C. and C.-J. Lin (2001). *LIBSVM: A library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Chen, B., J. Su, S. J. Pan, and C. L. Tan (2011). A Unified Event Coreference Resolution by Integrating Multiple Resolvers. In *IJCNLP*, pp. 102–110.

Chen, C. and V. Ng (2013). Chinese event coreference resolution: Understanding the state of the art. In *the 6th International Joint Conference on Natural Language Processing*, pp. 822—-828.

Chen, C. and V. Ng (2014). SinoCoreferencer : An End-to-End Chinese Event Coreference Resolver. In *LREC*, pp. 4532–4538.

Chen, C. and V. Ng (2015a). Chinese Event Coreference Resolution : An Unsupervised Probabilistic Model Rivaling Supervised Resolvers. In *NAACL HLT*, pp. 1097–1107.

Chen, C. and V. Ng (2015b, July). Chinese zero pronoun resolution: A joint unsupervised discourse-aware model rivaling state-of-the-art resolvers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 320–326.

Chen, C. and V. Ng (2016). Joint Inference over a Lightly Supervised Information Extraction Pipeline: Towards Event Coreference Resolution for Resource-Scarce Languages. In *AAAI*, pp. 2913–2920.

Chen, D., A. Fisch, J. Weston, and A. Bordes (2017). Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1870–1879.

Chen, Z. and H. Ji (2009). Graph-based event coreference resolution. In *TextGraphs-4*, pp. 54–57.

Chen, Z., H. Ji, and R. Haralick (2009). A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *eETTs*, pp. 17–22.

Choubey, P. K. and R. Huang (2017). Event Coreference Resolution by Iteratively Unfolding Inter-dependencies among Events. In *EMNLP*.

Choubey, P. K. and R. Huang (2018). Improving event coreference resolution by modeling correlations between event coreference chains and document topic structures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Volume 1, pp. 485–495.

Cybulska, A. and P. Vossen (2013). Semantic Relations between Events and their Time, Locations and Participants for Event Coreference Resolution. Number September, pp. 156–163.

Cybulska, A. and P. Vossen (2015). Translating Granularity of Event Slots into Features for Event Coreference Resolution. In *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT 2015*, pp. 1–10.

De Marneffe, M.-C., A. N. Rafferty, and C. D. Manning (2008). Finding Contradictions in Text. In *ACL-08 HLT*, pp. 1039–1047.

Denis, P. and J. Baldridge (2008). Specialized models and ranking for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 660–669.

Domingos, P. and D. Lowd (2009). *Markov Logic: An Interface Layer for Artificial Intelligence*. San Rafael, CA: Morgan & Claypool.

D'Souza, J. and V. Ng (2015a). Sieve-based entity linking for the biomedical domain. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 297–302.

D'Souza, J. and V. Ng (2015b). Sieve-based spatial relation extraction with expanding parse trees. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 758–768.

Duchi, J., E. Hazan, and Y. Singer (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*, 2121–2159.

Durrett, G. and D. Klein (2013). Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1971–1982.

Durrett, G. and D. Klein (2014). A joint model for entity analysis: coreference, typing, and linking. *Transactions of the Association for Computational Linguistics 2*, 477–490.

Ellis, J., J. Getman, D. Fore, N. Kuster, Z. Song, A. Bies, and S. Strassel (2015). Overview of linguistic resources for the TAC KBP 2015 evaluations: Methodologies and results. In *Proceedings of the TAC KBP 2015 Workshop*.

Ellis, J., J. Getman, N. Kuster, Z. Song, A. Bies, and S. Strassel (2016). Overview of linguistic resources for the TAC KBP 2016 evaluations: Methodologies and results. In *Proceedings of the TAC KBP 2016 Workshop*.

Fernandes, E. R., C. N. dos Santos, and R. L. Milidiú (2014). Latent trees for coreference resolution. *Computational Linguistics 40*(4), 801–835.

Getman, J., J. Ellis, Z. Song, J. Tracey, and S. Strassel (2017). Overview of linguistic resources for the TAC KBP 2017 evaluations: Methodologies and results. In *Proceedings of the TAC KBP 2017 Workshop*.

Gimpel, K. and N. A. Smith (2010). Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 733–736.

Gogate, V. and P. Domingos (2011). Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pp. 256–265.

Gong, Y., H. Luo, and J. Zhang (2018). Natural language inference over interaction space. In *Proceedings of the 6th International Conference on Learning Representations*.

Gurobi (2013). *Gurobi Optimizer Reference Manual*. Gurobi Inc.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural Computation 9*(8), 1735–1780.

Huang, Y. J., J. Lu, S. Kurohashi, and V. Ng (2019, June). Improving event coreference resolution by learning argument compatibility from unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, pp. 785–795. Association for Computational Linguistics.

Humphreys, K., R. Gaizauskas, and S. Azzam (1997). Event Coreference for Information Extraction. In *ANARESOLUTION*, pp. 75–81.

Ji, H. and R. Grishman (2011). Knowledge Base Population: Successful Approaches and Challenges. In *ACL*, pp. 1148–1158.

Jiang, S., Y. Li, T. Qin, Q. Meng, and B. Dong (2017a). SRCB Entity Discovery and Linking (EDL) and Event Nugget Systems for TAC 2017. In *TAC*.

Jiang, S., Y. Li, T. Qin, Q. Meng, and B. Dong (2017b). SRCB entity discovery and linking (EDL) and event nugget systems for TAC 2017. In *Proceedings of the TAC KBP 2017 Workshop*.

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Scholkopf and A. Smola (Eds.), *Advances in Kernel Methods - Support Vector Learning*, pp. 44–56. MIT Press.

Joshi, M., D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy (2020). SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics* (8), 64–77.

Joshi, M., O. Levy, L. Zettlemoyer, and D. Weld (2019, November). BERT for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 5803–5808.

Krause, S., F. Xu, H. Uszkoreit, and D. Weissenborn (2016). Event Linking with Sentential Features from Convolutional Neural Networks. In *CoNLL*, pp. 239–249.

Lee, H., A. Chang, Y. Peirsman, M. Surdeanu, and D. Jurafsky (2013). Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules. *Computational Linguistics*, 885–916.

Lee, H., M. Recasens, A. Chang, M. Surdeanu, and D. Jurafsky (2012). Joint Entity and Event Coreference Resolution across Documents. In *EMNLP-CoNLL 2012*, Number July, pp. 489–500.

Lee, K., L. He, M. Lewis, and L. Zettlemoyer (2017, September). End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 188–197.

Lee, K., L. He, and L. Zettlemoyer (2018, June). Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 687–692.

Li, P., G. Zhou, Q. Zhu, and L. Hou (2012). Employing compositional semantics and discourse consistency in chinese event extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1006–1016.

Li, W., M. Wu, Q. Lu, W. Xu, and C. Yuan (2006). Extractive Summarization using Inter- and Intra-Event Relevance. In *COLING-ACL*, pp. 369–376.

Liu, Z., J. Araki, E. Hovy, and T. Mitamura (2014a). Supervised Within-Document Event Coreference using Information Propagation. In *LREC*, pp. 4539–4544.

Liu, Z., J. Araki, E. Hovy, and T. Mitamura (2014b). Supervised within-document event coreference using information propagation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pp. 4539–4544.

Lu, J. and V. Ng (2016a). Event Coreference Resolution with Multi-Pass Sieves. In *LREC*.

Lu, J. and V. Ng (2016b). UTD's Event Nugget Detection and Coreference System at KBP 2016. In *TAC*.

Lu, J. and V. Ng (2017a). Joint Learning for Event Coreference Resolution. In *ACL*.

Lu, J. and V. Ng (2017b). Learning Antecedent Structures for Event Coreference Resolution. In *ICMLA*.

Lu, J. and V. Ng (2017c). UTD's Event Nugget Detection and Coreference System at KBP 2017. In *TAC*.

Lu, J. and V. Ng (2018). Event coreference resolution: a survey of two decades of research. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 5479–5486.

Lu, J. and V. Ng (2020). Event coreference resolution with non-local information. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pp. 653–663.

Lu, J. and V. Ng (2021). Constrained multi-task learning for event coreference resolution. In *Proceedings of Human Language Technologies: The 2021 Conference of the North American Chapter of the Association for Computational Linguistics*.

Lu, J., D. Venugopal, V. Gogate, and V. Ng (2016). Joint Inference for Event Coreference Resolution. In *COLING*.

Luan, Y., D. Wadden, L. He, A. Shah, M. Ostendorf, and H. Hajishirzi (2019). A general framework for information extraction using dynamic span graphs. In *Proceedings of NAACL*.

Luo, X. (2005). On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 25–32.

Manning, C. D., M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://www.cs.umass.edu/ mccallum/mallet.

McConky, K., R. Nagi, M. Sudit, and W. Hughes (2012). Improving event co-reference by context extraction and dynamic feature weighting. In *CogSIMA*, pp. 38–43.

Moosavi, N. S. and M. Strube (2016). Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 632–642.

Narayanan, S. and S. Harabagiu (2004). Question Answering Based on Semantic Structures. In *COLING*, pp. 693–701.

Ng, V. (2010). Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1396–1411.

Ng, V. and C. Cardie (2002a). Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 104–111.

Ng, V. and C. Cardie (2002b). Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (ACL-02), Philadelphia, PA, pp. 104–111. Association for Computational Linguistics.

Nguyen, T. H., K. Cho, and R. Grishman (2016, June). Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, pp. 300–309. Association for Computational Linguistics.

Nguyen, T. H., A. Meyers, and R. Grishman (2016). New York University 2016 System for KBP Event Nugget : A Deep Learning Approach. In *TAC*.

Pan, B., Y. Yang, Z. Zhao, Y. Zhuang, D. Cai, and X. He (2018). Discourse marker augmented network with reinforcement learning for natural language inference. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 989–999.

Parker, R., D. Graff, J. Kong, K. Chen, and K. Maeda (2009). English Gigaword fourth edition. In *Linguistic Data Consortium*.

Peng, H., Y. Song, and D. Roth (2016). Event Detection and Co-reference with Minimal Supervision. In *EMNLP*, pp. 392–402.

Pennington, J., R. Socher, and C. Manning (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Pradhan, S., X. Luo, M. Recasens, E. Hovy, V. Ng, and M. Strube (2014). Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 30–35.

Raghunathan, K., H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning (2010). A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 492–501.

Ramage, D., D. Hall, R. Nallapati, and C. D. Manning (2009). Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, Stroudsburg, PA, USA, pp. 248–256. Association for Computational Linguistics.

Recasens, M. and E. Hovy (2011). BLANC: implementing the Rand Index for coreference evaluation. *Natural Language Engineering 17*(4), 485–510.

Sangeetha, S. and M. Arock (2012). Event coreference resolution using mincut based graph clustering. In *Proceedings of the 4th International Workshop on Computer Networks & Communications*, pp. 253–260.

Song, Z., A. Bies, S. Strassel, T. Riese, J. Mott, J. Ellis, J. Wright, S. Kulick, N. Ryant, and X. Ma (2015). From light to rich ere: Annotation of entities, relations, and events. In *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT*, pp. 89–98.

Soon, W. M., H. T. Ng, D. Chung, and Y. Lim (2001). A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics 27*(4), 521–544.

Turian, J., L. Ratinov, and Y. Bengio (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 384–394.

Vilain, M., J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman (1995). A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pp. 45–52.

Wiseman, S., A. M. Rush, S. Shieber, and J. Weston (2015a, July). Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1416–1426.

Wiseman, S., A. M. Rush, S. M. Shieber, and J. Weston (2015b). Learning Anaphoricity and Antecedent Ranking Features for Coreference Resolution. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1416–1426.

Xue, N. and M. Palmer (2009). Adding semantic roles to the Chinese Treebank. *Natural Language Engineering 15*(1), 143–172.

Yang, B., C. Cardie, and P. Frazier (2015). A hierarchical distance-dependent Bayesian model for event coreference resolution. *Transactions of the Association for Computational Linguistics 3*, 517–528.

Yang, Z., B. Dhingra, Y. Yuan, J. Hu, W. W. Cohen, and R. Salakhutdinov (2017). Words or Characters? Fine-grained gating for reading comprehension. In *Proceedings of the 5th International Conference on Learning Representations*.

BIOGRAPHICAL SKETCH

Jing Lu obtained her Master of Science in Computer Science in 2020 from The University of Texas at Dallas, her Bachelor of Engineering in Electronic Information Engineering in 2012 from the University of Electronic Science and Technology of China and her Bachelor of Science in Electrical Engineering in 2012 from the University of Missouri. Jing's areas of interest are natural language processing, machine learning and information retrieval.

CURRICULUM VITAE

Jing Lu

Educational History:

Ph.D., Computer Science, University of Texas at Dallas, 2021
M.S., Computer Science, University of Texas at Dallas, 2020
B.E., Electronic Information Engineering, University of Electronic Science and Technology of China, 2012
B.S., Electrical Engineering, University of Missouri, 2012

Industry Experience:

Research Intern, Google AI, May 2020 - Dec. 2020

- Build neural text retrieval systems based on dual encoders for open-domain question answering tasks.
- Implement hard negative mining strategies to improve model performance.
- Outperform traditional sparse information retrieval models (e.g. BM25) and recent neural models.
- Work collaboratively with the teams in Mountain View and London.

Research Intern, Google AI, May 2019 - Aug. 2019

- Build a cross-attention model based on BERT for cross-document entity coreference resolution.

Research Experience:

Research Assistant, University of Texas at Dallas, Fall 2015 - Present

- Event Detection and Coreference Resolution
  - Develop neural models and statistical models for jointly learning event coreference related tasks.
  - Achieved the state-of-the-art performance on English and Chinese benchmark datasets.
- Language-Agnostic Political Event-Data Coder

  - Develop a rule-based event coder for political and social events in CAMEO ontology based on universal dependencies.
  - Extend the English-only approach to code events from Spanish and Arabic texts.
- Automated Analysis of Bug Descriptions
  - Design and implement an automated approach to detect the absence (or presence) of the expected behavior and the steps to reproduce in bug descriptions.
  - Develop a neural sequence labeling model to identify the steps to reproduce in bug descriptions for GUI-based Android apps and an automated approach to assess the quality of them.

Visiting Scholar, University of Kyoto, June 2017 - July 2017

Awards & Honors:

Ericsson Graduate Fellowship, University of Texas at Dallas 2017-2018, 2018-2019, 2019-2020
The First-Class People's Scholarship Sept. 2010
National Scholarship, Ministry of Education of the P.R. China (Top 0.2%) Sept. 2009