Abstract of “Cross-Document Coreference Resolution for Entities and Events” by Chris Tanner, Ph.D., Brown University, May 2019.

Coreference Resolution is a fundamental natural language processing (NLP) problem, as it attempts to resolve which underlying discourse objects refer to one another. Further, it serves as an essential component of many other core NLP tasks, including information extraction, question-answering, document summarization, etc. However, decades of research have primarily focused on resolving *entities* (e.g., people, locations, organizations), with significantly less attention given to *events* — the actions performed. This dissertation focuses on improving event coreference. We first detail existing research, while addressing potential weaknesses in current systems: the reliance on dozens of hand-engineered lexical features, and agglomerative-based clustering that is limited to mention-to-mention comparisons. We develop a state-the-art relational-based model which uses almost no features, along with a neural clustering approach that is more holistic than existing mention-based clustering approaches. Last, we research the benefits of including entity information and further this by resolving both entities and events. Our model is novel in demonstrating a symbiotic relationship.

Cross-Document Coreference Resolution for Entities and Events

by

Chris Tanner

B. S., Computer Science, Florida Institute of Technology, 2006

B. S., Applied Mathematics, Florida Institute of Technology, 2006

M. S., Computer Science, UCLA, 2009

Sc. M., Brown University, 2015

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2019

This dissertation by Chris Tanner is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____
Eugene Charniak, Director

Recommended to the Graduate Council

Date _____ _____
Ellie Pavlick, Reader
Brown University

Date _____ _____
Stefanie Tellex, Reader
Brown University

Date _____ _____
Michael Littman, Reader
Brown University

Approved by the Graduate Council

Date _____ _____
Andrew G. Campbell
Dean of the Graduate School

Acknowledgements

"No [person] is an island."

— John Donne (1624)

I would like to thank Brown University's Computer Science department for establishing and maintaining an incredibly caring, transparent environment and familial community. I thank my adviser, Eugene Charniak, for giving me unlimited freedom to pursue any research topic. The level of independence allowed me a healthy amount of room to have my own successes and failures, providing memorable lessons and growth. Eugene was incredibly available, always with an open door and everlasting curiosity and eagerness to discuss research ideas; I will miss our brainstorming sessions and his world-renowned expertise and intuition. My committee members, Ellie Pavlick, Stefanie Tellex, and Michael Littman, served as great resources and lifelines.

I thank fellow labmates, Do Kook Choe (D.K.), Ben Swanson, Rebecca Mason, and Byron Wallace for their support and helpful comments, especially during my early years in the program. Further, I was fortunate to work at three research internships during my Ph.D. time, each of which provided priceless experiences. I would like to especially thank: Ben Van Durme, Chris Callison-Burch, Mark Drezde, and Eugene Agichtein from Johns Hopkins HLTCOE SCALE; Jiri Navratil from IBM Watson; and Ben Lambert, Rohit Kumar, and David Murgatroyd from Spotify.

I am incredibly grateful for having the best network of friends anyone could ask for: John Hutchinson, Mark Toole, Sean Householder, Jeremy Putman, Brandon Williams, Michaeel Kazi, Dan Hendrickson, Mia Molero, George Francis Jr, Brian Kaemingk, Bryce Richards, Rici Hammer, Charlie Dagli, Renee Smith, Nedi Daskalova, Alexandra Papoutsaki, Evgenios Kornaropoulos, Patrick Heck, Emily Hollenbeck, Jeff Rasley, Jon Mace, Stephen Brawner, Ben Abbatematteo, Kavosh Asadi, Jana Jarolimova, and many others. Their support and our fun times together mean the world to me. Finally, I thank my parents for exemplifying the values of hard work, and for their incredible, unbounded efforts to provide for me during childhood.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Coreference resolution is the task of identifying, within a body of text, which words refer to the same underlying discourse objects. For example, if a sentence read, "José voted for the Senator Warren because her values best aligned with his," a human reader would easily understand that the words "Senator Warren" and "her" refer to the same person, whereas "José" and "his" both refer to a different entity. While this task is usually intuitive for humans, it is difficult for computers. Yet, understanding who is whom and what is what, within a body of text, is a crucial component for any system that aims to understand discourse and the meaning of text. That is, having knowledge of the participants involved and the actions performed is imperative for providing a holistic view of any narrative. Naturally, coreference resolution is not only a fundamental task within the field of natural language processing (NLP), but it is also useful when applied toward other NLP tasks such as information extraction [49], question answering [76], topic detection [2], summarization [26], and more. As a simple example, if one performs a Web search for "Justin Trudeau," some of the search results will contain sentences which only refer to him as "Prime Minister," "he," or "Trudeau," and correctly using this information is essential for returning information that is relevant to the user's query. Natural language systems continue to play an increasingly large role in our daily lives, for example: voice assistants such as Google Home, Amazon's Alexa, and Apple's Siri; personalized search results; automated chatbots serving as customer service agents; and auto-complete suggestions for writing emails. Consequently, researching and improving coreference resolution systems is useful from both academic and industrial purposes, as it is a complex computational problem that can yield profound real-world impact.

1.2 Terminology

This dissertation explores several components of coreference resolution, so we now define the related, key terminology that will be used throughout.

A **mention** is a particular word, or group of words, in a document which represents an underlying *entity* or *event*, such as *Justin Trudeau*, *he*, or *spoke*. Coreference resolution is concerned with determining which *mentions* co-refer to one another. Using our original example sentence about Justin Trudeau, words such as "with," "today," and "about" are not mentions and thus coreference systems do not attempt to resolve them.

An **entity** may be a person, location, time, or an organization. The mentions which refer to them may be *named*, *nominal*, or *pronominal*:

- Named mentions are represented by proper nouns (e.g., André Benjamin or Pakse, Laos)
- Pronominal mentions are represented by pronouns (e.g., she or it)
- Nominal mentions are represented by descriptive words, not composed entirely of a named entity or pronouns (e.g., The *well-spoken citizen*)

An **event** can generally be thought of as a specific action. Quine [86] proposed that an event mention refers to an action that is grounded to a specific time and location, and that two event mentions are co-referent if they share the same spatiotemporal location. This definition[1] has become the general consensus within the community, and we adopt it, too. Specifically, two co-referent events must share the same *properties* and *participants*, which must be at least reasonably implied and not explicitly contradicted. For example, in Figure 1.1, the first sentence in document #1 and the first sentence in document #2 contain co-referent events ("checked into" and "check into"), but neither of those event mentions co-refer with any additional mentions. Often times, the participants may be referred to in different ways, implied, or missing altogether.

When two mentions are determined to be co-referent with each other, we can refer to them as being **linked** together. Coreference resolution is concerned with linking either entities together and/or events together; that is, entities shall not be linked to events, and doing so would be considered an incorrect link. Although one may be interested in evaluating coreference systems by their ability to correctly link *pairs* of mentions [106], coreference resolution is ultimately a clustering task, whereby we wish to group all like-mentions together, as shown with colored boxes in Figure 1.1. Specifically, coreference systems aim to find a globally-optimal fit of mentions to clusters, whereby every mention m in the corpus is assigned to exactly one cluster C, such that

[1] Hovy, et al. [48] provide an in-depth study of varying definitions.

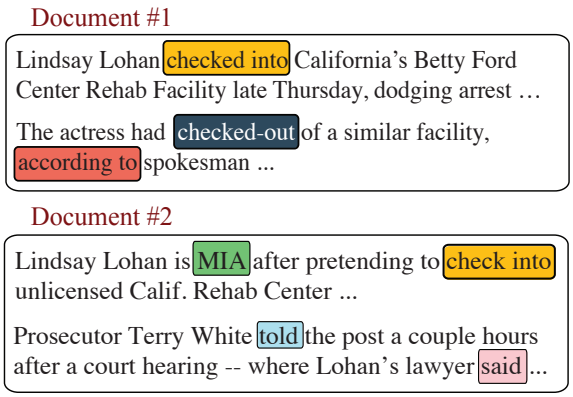


Figure 1.1: Sample of a coreference resolution corpus (ECB+), depicting gold coref mentions as having shared box colors.

every $m_i, m_j \in C$ are co-referent with each other. If a given m_i is not coreferent with any other m_j, then it should belong to its own cluster C with a membership of one.

Given a corpus of text documents, coreference resolution can be trained and evaluated on either a **within-document** or **cross-document** basis:

- **Within-document** is when each mention may only link to either (1) no other mention; or (2) other mentions which are contained in the same document. Even if the gold truth data denotes mentions from differing documents are co-referent, this cross-document data is not used for training or testing.
- **Cross-document** is when the entire corpus is available for linking; a mention is eligible to be co-referent with mentions in any other document, and the evaluation reflects the same. As described in [100], cross-document evaluation is normally conducted by transforming the entire corpus into a single "meta-document," whereby all documents are simply concatenated together.

1.3 Coreference Systems

Coreference systems are predicated upon knowing which words constitute mentions. This process of identifying entity/event mentions is the focus of a separate, distinct[2] task called *mention detection*. The identified mentions are used by coreference resolution models.

[2]Only recently have there been new efforts to combine both mention detection and coreference resolution [61, 111].

1.3.1 Mention Detection

Mention detection has remained a fundamental NLP task for several decades [75]. Identifying mentions is useful not only for coreference resolution but also other NLP tasks such as information retrieval, relation extraction [64], and entity disambiguation [23]. When mention detection concerns entities (as opposed to events), research is commonly conducted for the task of *named entity recognition*, which not only detects the mention boundaries but classifies/labels each mention as being one of several possible class types (e.g., person, organization, location).

Named Entity Recognition

The earliest work started in 1991 with the task of identifying company names [89]. In 1996, the MUC-6 conference [42] focused on Information Extraction tasks, which included coining the phrase "named entity" and drastically increasing attention to mention detection. Early work demonstrated state-of-the-art performance with Hidden Markov Models (HMMs) [11] and Conditional Random Fields (CRFs) [72]. Presently, the best performing systems use neural sequence-based approaches, such as Bidirectional LSTMs [18, 47], Convolution Neural Networks (CNNs) [69], and character-based LSTMs [1].

Event Detection

Event detection has received less research attention than named entity recognition. Recent state-of-the-art performances are achieved with Bi-directional LSTMs [77], CNNs [35], and Global Context Layers (GCL) [90]. Event detection is also used in conjunction with Semantic Role Labelling (SRL) [81], which addresses a similar and more encompassing problem. Specifically, SRL is a shallow semantic parsing task, whereby the goal is to identify each predicate in a sentence, along with its constituents and how they fill a semantic role (e.g., Agent, Patient, Instrument, etc) [39, 85]. For example, if a sentence were "Mary sold the book to John," an SRL system should determine that "to sell" is the predicate (i.e., action), "Mary" is the seller (i.e., Agent), "the book" is the object (i.e., theme) and "John" is the receiver (i.e., recipient). SRL systems and Event Detection systems both often rely on using many lexical and syntactical features, including those from constituency parsers [99], dependency parsers [50], etc.

1.3.2 Coreference Resolution

As mentioned, coreference systems aim to create the correct clusters of mentions; however, due to the large number of possible combinations, finding a globally-optimal assignment of clusters is NP-Hard and thus computationally intractable. In an attempt to avoid this, systems typically perform pairwise-mention predictions, then use those predictions to build clusters. The specific modelling strategies for such approximately fall into the following categories: (1) mention-ranking / mention-pairs; and (2) entity-level / event-level, as described below. Each modelling paradigm offers its own strengths and weaknesses, and consequently, state-of-the-art results are often achieved from either approach; no single paradigm is unanimously better than the other.

Mention-ranking models define a scoring function $f(m_i, m_j)$ which operates on a mention m_j and possible antecedent m_i, where m_i occurs earlier in the document and could be *null* (represented by ϵ and denoting that m_j is not coreferent with any other mention); e.g., Wiseman, et al.'s [105]. These models aim to find the ideal m_i antecedent for every m_j mention. After every mention has decided to link to ϵ or a previous mention, it is common practice to define each cluster simply by joining together all mentions which are connected by a single path. For example, if m_3 predicts m_2 as its antecedent, and m_2 predicts m_1 as its antecedent, then $\{m_1, m2, m3\}$ are all connected. This assumes that the transitive property holds true, which is a potential weakness, as there is no direct, holistic consideration given to every unique mention within a particular cluster.

Mention-pair models score all pairs (m_i, m_j), in contrast to mention-ranking models which aim to find the ideal m_i antecedent for every m_j. After every pair of mentions has been scored, it is common practice to cluster mentions in an iterative fashion, one mention at a time. This approach is commonly referred to as *best-first* or *easy-first* since each decision is based on a single "lowest cost" score (a la agglomerative clustering). Because mention-pair models base their predictions on the information from just two mentions at a time, they are by definition less expressive than entity/event-level models. Yet, their inference can be relatively simple and effective, allowing them to be fast and scalable. Consequently, they have often been the approach used by many state-of-the-art systems [31, 97].

Entity/Event-level Instead of operating on a per-mention basis, these models differ in that they focus on building a global representation of each underlying entity or event, the basis of which determines each mention's membership [22, 104]. These models are attractive due to the intuitive nature of modelling each entity with its own representation; however, challenges include (1) deciding how to represent each entity as it is being developed; (2) decided how many entities to model.

1.4 Contributions

In this dissertation, we research event coreference resolution for both within-document and cross-document settings. We summarize existing research while identifying what we believe to be weaknesses. We address the plethora of features that are commonly used in others' systems, and we offer a state-of-the-art mention-pair neural model (Conjoined CNN) that uses very few features (lemma embeddings and character embeddings). While most coreference models perform agglomerative clustering on their predicted pairs of mentions, we improve this by offering a neural clustering model that takes a more holistic approach. After carefully reviewing our overall system's performance, we focus on how to better use mentions' context. Specifically, we use structured, dependency tree representations of sentences, allowing us to glean and use useful representations of entities. These entity embeddings, when combined with our original event coreference model, offer the best performance yet. Further, the reverse holds true, too: using tree-based event representations, when coupled with our original entity coreference model, yields improved results. Thus, we demonstrate a symbiotic relationship by combining entities and events to achieve better results than that from any individual model.

Chapter 2

Background

Coreference resolution has a long, rich history of research. Although event coreference has received significantly less research attention than entity coreference, there has been continued interest ever since it was formally introduced in the late-1990s. Both lines of research make heavy use of machine learning. Consequently, this chapter first provides a succinct explanation of the relevant machine learning topics, including neural networks (i.e., deep learning), then briefly describes two representations that are commonly used within NLP: word embeddings and dependency parses. Last, we provide an overview of coreference resolution work.

We assume the reader has background knowledge in probability, statistics, linear algebra, and calculus. While our aim is for this dissertation to be self-contained, providing a comprehensive expository of machine learning, deep learning, and natural language processing (NLP) is beyond our scope. For those who desire such information, we refer the interested reader to the excellent textbooks by Bishop [12], Jurafsky and Martin [51], Goodfellow et al. [40], and Charniak [15].

2.1 Machine Learning

Machine Learning is a large sub-field of computer science that concerns developing models that learn to perform a given task, for a specific set of data. Examples include predicting numeric values (e.g., stock prices), predicting categorical choices (e.g., the tumor is benign; the object in the road is a bicyclist), clustering data (e.g., grouping similar users together based on their movie-watching preferences). There is a plethora of different learning objectives, environment scenarios, and types of data. Despite this enormous range of aspects and approaches to machine learning, there are two large paradigms of algorithms that span a rich set of

models: *supervised learning* and *unsupervised learning*. Together, these two categories encompass a majority of machine learning models.

Supervised learning is when a model uses data that is labelled/annotated with the desired output. This can be viewed as learning a function that maps a set of inputs to a desired set of outputs (e.g., making predictions for a certain task). Unsupervised learning is when a model uses data that is unlabelled and not annotated with our desired objective (e.g., clustering similar users together).

As an illustrative example: all machine learning models rely on using data. When that data is provided to a model, it is referred to as *input data*, and each instance of data may contain many aspects/features. For example, say we have a dataset corresponding to weather. Perhaps we have five years worth of data, and each instance/example represents a single day of weather. Further, say each day's data includes four aspects/features: (1) did it rain yesterday (value of 0 or 1); (2) is the humidity greater than 70% (value of 0 or 1); (3) is the temperature greater than 80° F (value of 0 or 1); and (4) is the sun visible (value of 0 or 1). An example of unsupervised learning is if our task is to find patterns or groups of days with similar weather, perhaps over time. However, a supervised task could be to predict if each day will encounter rain. In this scenario, our data would need to also include the gold truth information of if it actually rained on each day. The training phase of our supervised model would use this gold, labelled information. However, at testing time, our model would only have access to the original features, and its goal is to predict the correct, labelled answer. Note: it is imperative for the training and testing data to contain disjoint data samples (i.e., days of weather); otherwise, the model will perform falsely well, akin to a student cheating on an exam by having previously seen the exact exam questions (testing data).

For the entirety of this dissertation, the models we develop are supervised approaches. Specifically, we develop neural network models. We now provide a brief overview of neural network models.

2.2 Neural Networks and Deep Learning

Neural networks are machine learning models that were roughly inspired by information processing and communication patterns in biological systems — the vast interconnected network of neurons within brains [34, 80]. In 1943, McCollough and Pitts [36] first developed the idea of a computational "neuron" — it simply allows for multiple input values x_i (each of which is a 0 or 1) and produces an output value $\hat{y}$ that is 0 or 1 (representing the neuron is quiescent or excited, respectively). This was extended by the concept of the *perceptron*, a breakthrough in 1958 by Rosenblatt.

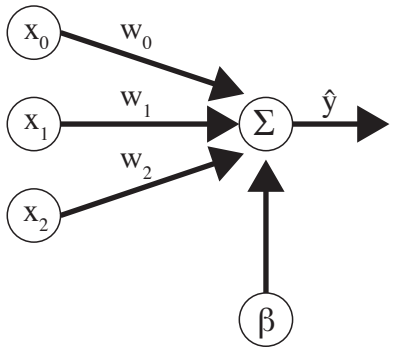


Figure 2.1: A perceptron, the fundamental building block of neural networks. Arrows denote that the values at the tail are inputs into the item at the head.

A perceptron (Figure 2.1) consists of multiple input values x_i, each of which can be of any value (limited only by the developer) and is multiplied by a weight value w_i — which is also a continuous-valued number. We then take the sum of all the weighted values and add a bias β. If this total is above 0, our perceptron emits $\hat{y}$ as 1; otherwise, it outputs $\hat{y}$ as being 0, as depicted by:

$$f(x) = \begin{cases} 1, & \text{if} \quad \sum_i w_i x_i + \beta \geq 0. \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Note, the perceptron is merely a basic function, effectively identical to the canonical $y = mx + \beta$ from algebra. Yet, despite this simplicity, the perceptron model already allows one to make predictions for particular tasks: initially, all w_i weight values and the β bias are set to random values. Then, for any given set of input values (i.e., features, such as the four weather features listed above), the model will multiply the input by the w_i weights and ultimately emit a 0 or 1, which we refer to as $\hat{y}$. Based on the desired output y, the perceptron's output $\hat{y}$ will either be right or wrong. If it is wrong, the w_i weights and bias β should be updated so as to make the $\hat{y}$ output closer to the correct output y. If the program is given many data examples (training data), the hope is that the perceptron will learn appropriate weights such that later, when presented with unseen testing data, the model will produce correct predictions/classifications. While we do not detail the exact process of updating the weights and bias, we refer the interested reader to [15].

Further, if one were interested in modelling the likelihood of not only one class/output (e.g., likelihood of rain), but multiple, possible classes/outcomes (e.g., likelihood of rain, likelihood of snow, etc), one could use a perceptron for each class, as shown in Figure 2.2. Each perceptron's output corresponds to its prediction for that class. Thus, the program predicts that the most likely outcome is the perceptron class that has the highest

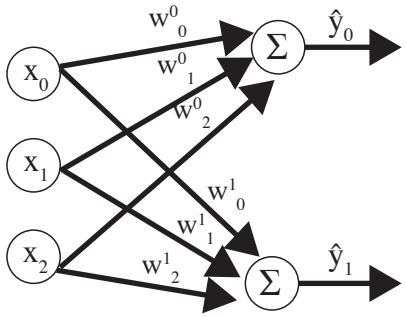


Figure 2.2: A multi-class perceptron. Each $\hat{y}_i$ class has its own bias β_i, yet we omit it here for readability.

score. In this scenario, instead of using Equation 2.1, it is standard practice to calculate a probability for each of the K outcomes. This is done by the *softmax* function (Equation 2.2), which normalizes every score via dividing each by the total sum of outcomes. Note, even if the original $\hat{y}_i$ value were negative, taking the $e^{\hat{y}_i}$ makes it positive.

$$P(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_i^K e^{\hat{y}_i}} \tag{2.2}$$

Let each of our input data samples have N input values (i.e., $x_0, x_1, ...x_{n-1}$), then one could interpret each datum as existing in N-dimensional space. If we were to plot all of our training data in this space, ideally there would exist a single decision surface which would perfectly separate our different classes of data (e.g., rained or not). If this is the case, our data is said to be *linearly separable.* Unfortunately, our aforementioned examples of single-perceptrons and multi-class perceptrons have only had one layer of outputs. That is, each perceptron yields our final output/predictions. While these models may be useful for a range of tasks, they are ultimately limited in their expressive power, for they can only accurately classify linearly separable data [44].

To remedy this limitation, one can stack multiple perceptrons after one another; thus, the output of one perceptron serves as the input into another, subsequent perceptron, which then ultimately produces a final output. The intermediate perceptron is referred to as being a *hidden unit*, and its actual value is determined by an *activation function* — the purpose of which is to scale the value so that it is within a reasonable range (e.g., 0 to 1), since it will serve as input to the next layer. Common activation functions include the *sigmoid function* (Equation 2.3), Rectified Linear Units (ReLU) (Equation 2.4), and hyperbolic tangent (tanh). If the neural network contains multiple hidden perceptrons, each of which serves as input to their own respective, subsequent perceptron, then the model is said to have a *hidden layer* (Figure 2.3).

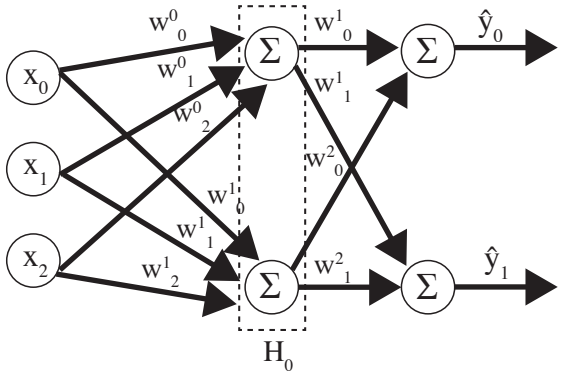


Figure 2.3: A multi-class neural network with a hidden layer, denoted by the dashed box.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

$$\sigma(x) = max(0, x) \tag{2.4}$$

The networks we have presented so far are called *feed-forward neural networks*. There are many other variants, but all neural models start with input data, then subsequently transform the data by a series of functions (i.e., activation functions), and ultimately output a value(s) pertaining to the task at hand. While the mathematical details are not important for now, the key point is that a neural model uses a multitude of weighted data transformations as its core computing mechanism. If the network has multiple hidden layers, it is called a deep neural network, and the overall framework is referred to as *deep learning*.

It is common for deep neural models to contain thousands of non-linear transformations/functions, and since the model at large accepts inputs and emits an output, one can view the entire neural network model as simply one large, non-linear function. Deep Learning has effectively impacted every area of artificial intelligence and machine learning. In fact, it has advanced the state-of-the-art performance in nearly every area of NLP, from machine translation [16, 33], language modelling [27, 83], constituency parsing [19, 56], question answering [28, 94], and image captioning [103, 107], just to name a few. While it is beyond the scope of this work to include a comprehensive foundation in deep learning, we refer interested readers to the introductory textbooks by Goodfellow, et al. [40] and Charniak [15].

2.2.1 Loss Functions

Neural networks work due to their learned weights. That is, during the model's training phase, weight parameters are learned by being iteratively adjusted such that the model performs well on the training data for a given task. The metric that determines how well/poorly the model performs at the given task is defined by the *loss function*. Specifically, a loss function produces a single scalar value that represents how inaccurate the model's predictions were compared to the golden, correct labels. It is up to the developer to determine which loss function to use, and commonly used ones include cross-entropy for multi-class categorical predictions and mean-squared error for linear regression.

Once a model calculates its loss, it needs to update its weight parameters accordingly. This is done via *backpropagation*, which is a core component of neural networks, and the details are beyond the scope of this dissertation. We refer the interested reader to [93]. Independent of backpropagation is the component of optimization, which determines the manner and degree to which each weight is updated. This affects how quickly/slowly the model learns weights. Examples of optimization include the Adagrad [30] and Adam [55] algorithms.

2.2.2 Convolutional Neural Networks (CNNs)

So far, all of the neural networks we have discussed have immediately used the *all* of the current layer's inputs for computing the next layer of values. However, in some applications, it can be largely beneficial to focus on sub-regions of inputs. For example, within the field of vision, our inputs are often two-dimensional images, represented as a matrix of continued-valued pixel values (typically scaled from their original 0-255 pixel color value to be between zero and one). For example, in Figure 2.4, the input is a picture of a hand-written digit, two, which is part of the famous MNIST [59] dataset.

If the task for our program were to accept the hand-written image and predict what the digit is (zero through nine), one could use a feed-forward neural network, like ones we discussed. In particular, the input layer could have one neuron for each of the 784 pixels in the 28x28 image. This would be highly sensitive to input changes though; imagine, if our example image of a two were slightly moved to the left and down by a few pixels, it would cause nearly every input neuron to be different. Or, imagine if the intensity of the image changed ever so slightly (e.g., if the person who wrote the digit pressed harder than before), every neuron would change, which would make learning difficult. Alternatively, it would be best if our model were robust to small changes in the inputs, were better at recognizing local patterns (e.g., there is a swirly characteristic

in the lower-left region), and could use those patterns for making global predictions (e.g., the digit is a two). Looking at small, regional patterns is more robust than directly using every individual input pixel because regions, relative to one another, will likely behave similarly across all images of a given digit. A convolutional neural network (CNN) is designed to address these exact issues. Here, we provide a succinct overview of how they work, and we refer the interested reader to more thorough explanations [15, 52, 113].

A CNN learns regional patterns by using a *convolutional kernel* (also called *convolutional filter*), which is a two-dimensional weight matrix that is smaller in size than the original input (e.g., a size of 3x3 is common). As depicted in Figure 2.4, the convolutional kernel is repeatedly multiplied (dot product) by successive regions of the original input, in a sliding window fashion. For any one particular instance of multiplying the kernel by the input, we use an activation function, which outputs a single scalar value. Thus, after all successive applications of the kernel have been multiplied against the original input, we will have a one-dimensional array/vector of scalar values called a *feature map*. We compress this information into the most meaningful bits via a process called *pooling*. One of the most common pooling techniques is *MaxPooling*, which simply emits the maximum value from the vector. We are now left with a single scalar again.

Instead of using only one convolutional filter (a matrix of weights that is learned), one can simultaneously apply N different convolutional filters. Thus, instead of resulting in just a single scalar value from having performed pooling, we are left with a vector of length N. This vector can then be used to make a final prediction towards the goal (e.g., which digit is it?). In keeping with all other neural networks, based on our objective function, our weights are updated via backpropagation.

There are many variants and nuanced elements to this outlined CNN approach, such as optionally performing this convolutional process multiple times (stacked) before making the final prediction, deciding how to slide the convolutional kernel across the input (e.g., sliding it over one pixel at a time, or skip a few pixels, and when to stop sliding, etc.), selecting the activation function, bias, and initialization, the size of the convolutional kernel, etc. Nonetheless, the overall process remains the same. Further, instead of only operating on two-dimensional images, it is common to use CNNs for natural language processing, too. In this scenario, one often represents each word (e.g., word embedding) as its own row in the input matrix, usually in the linear fashion the words appeared in the corpus. The columns correspond to individual features for each word. For example, if one were modelling a word its context (three words that appear before and after it), where each word is represented by a 300-length word embedding, then the matrix would be of size $7x300$.

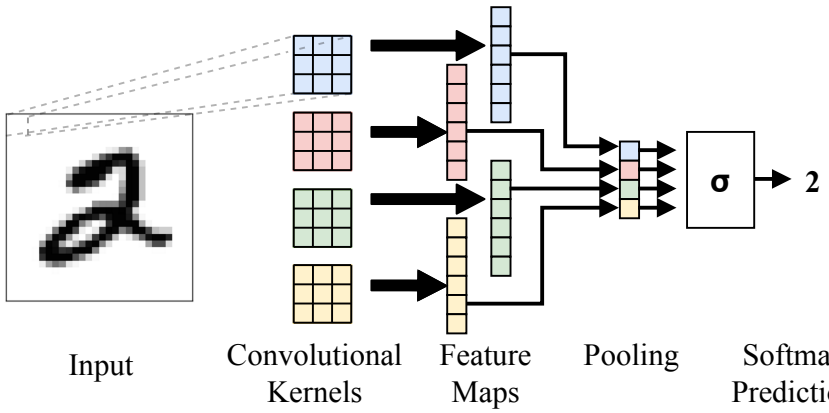


Figure 2.4: The canonical architecture for a Convolutional Neural Network (CNN).

2.3 Language Representations

First, we describe the difference between a *word token* and a *word type*. A *word token* refers to a specific instance of a word that is used in text, whereas a *word type* refers to its global existence/representation. For example, if a sentence were "Run Spot, run", there are three word tokens (i.e. run, spot, run), but only two word types (i.e., run, spot).

2.3.1 Word Embeddings

Representing words is a necessity for all of NLP, regardless of the task at hand. Since words are the primary ingredient and main currency in NLP, the representations thereof naturally play a large role. The seminal research of word embeddings can be traced back to the vector space model by Salton et al. in 1975 [91]. This was one of the early works that popularized attempts to represent each word (or document) as a fixed-length vector. There are many approaches to learn word representations, from distributional representations, clustering-based representations, and distributed representations, with the latter being the most popular and successful. Specifically, distributed representations aim to represent each word as a fixed-length, low-dimensional (e.g., 100-400 dimensions), continuous-valued vector. Each dimension of the embedding represents a latent feature of the word, ideally capturing the most important syntactic and semantic properties.

In the early 2000s, researchers began to use neural networks to create word embeddings, largely due to Bengio et al. [7] work in developing the first *large-scale* neural network language model. Most impactful was the *word2vec* model by Mikolov et al. [73] in 2013. In short, word2vec, and many other models, learns embeddings for each word by considering words' contexts and co-occurrence patterns. One can use different

learning objectives to create the embeddings, such as trying to predict each given word by merely looking at its neighboring context — the continuous bag-of-words (CBOW) method. Another highly popular algorithm, which we use for this dissertation, is GloVe [82]. In short, GloVe aims to construct word embeddings that capture meaning by using word co-occurrence *ratios*, rather than the raw counts of words. It is possible to run the GloVe algorithm either on one's own corpus (such as the corpus we use), or to use pre-computed word embeddings after having run GloVe on very large (840-billion word corpora) — which the GloVe authors provide for free use. One of the benefits of the former approach is that the embeddings are relevant to one's actual dataset; one of the benefits of the latter is that it has more data to learn from, although the domain may differ. Regardless, a property of all modern word embeddings (e.g., word2vec and GloVe) is that words that are more similar to one another will ideally have embeddings that proportionally reflect such. Both word2vec and GloVe construct embeddings for *word types*, yet many modern approaches also do so for individual *word tokens*, since specific context matters.

2.3.2 Recurrent Neural Networks (RNNs)

In addition to the previously mentioned neural network architectures (feed-forward and convolutional), recurrent neural networks are also an incredibly popular architecture. While they may be used for a wide-variety of tasks, we will provide a brief overview of them by concerning the task of language modelling. As a reminder, language modelling concerns computing a probability distribution for any sequence of words (e.g., a sentence). In doing so, it has the ability to calculate the most probable next word, given some prior context.

One approach to predicting the next word of a sequence is to use a feed-forward neural network: the input could be the N words of the sequence so far, and the target output is the next word. A serious issue is that each prediction is treated independently from the previous — as if we are starting from scratch on each successive input, despite the fact that the input differs from the previous input by only one word. Thus, there is no concept of maintaining a current *state* as we predict each word. RNNs address this issue by basing each prediction not only on its current input but also on the hidden/latent state from the previous prediction, as depicted in Figure 2.5 by the h_i blue, hidden state boxes. This feedback loop of information gives rise to the *recurrent* part of the network's name.

While RNNs often perform well for tasks that involve sequential information, like language modelling, they have limitations in factoring in long-range dependencies [9, 46]. Long-short Term Memory Networks (LSTMs) [47] are a variant of RNNs that address this issue by adding "gating" units to the network, effectively allowing the model to selectively "forget" and "remember" certain elements of the information as it is sequentially

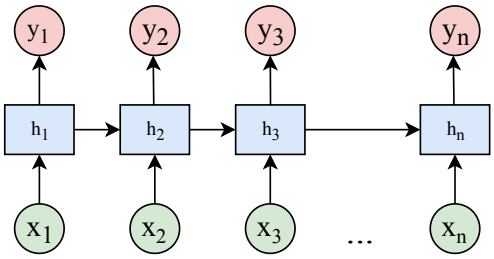
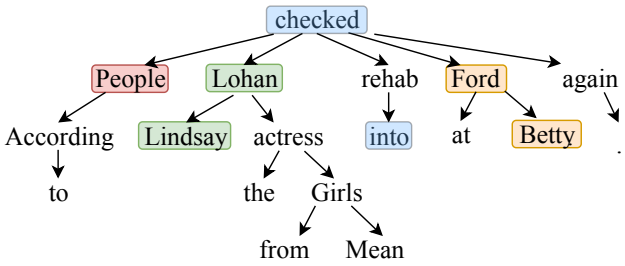


Figure 2.5: A Recurrent Neural Network (RNN) architecture. x's represent the input, such as "The dog ran fast" could be values for x_1, x_2, x_3, x_4, respectively. h's represent the hidden states, and y's represent the predicted output values.



According to People, Lindsay Lohan, the actress from Mean Girls, checked into rehab at Betty Ford again.

Figure 2.6: An example dependency parse. Each arrow's tail corresponds to the governing word, and the arrowhead corresponds to the word that depends on it, via the listed relation.

processed through the network. Currently, LSTM-based models are the de facto standard for processing sequential information. For a more thorough explanation and foundation, we recommend [15, 40].

2.3.3 Dependency Parsing

Although we, as humans, represent and communicate language as a sequence of words, the underlying structure is compositional. Dependency Parsing is one of the useful ways to compose a sentence by its words. Specifically, it determines which words depend on which other words, and it explicitly labels the relationship, too. For example, in a simple sentence of *The dog ran fast*, one dependency relation is to identify that *dog* depends on the verb *ran*, and its relation is *nsubj* — illustrating that it is serving as the subject to the verb. Figure 2.6 illustrates the dependency parse for an example sentence, yet for readability, we omit the relations (which would be listed on the arrow connections between each word). Dependency Parses have been shown to be highly useful for a range of tasks, especially those concerning natural language understanding [3, 13, 63].

2.4 Coreference Resolution

The seminal research can be traced back to the MUC conferences which focused on entity-based coreference for limited scenarios such as terrorist attacks, plane crashes, resignations, etc. [5, 49]. Since this beginning, coreference systems have used a wide variety of models, and we now summarize some of the most impactful systems.

2.4.1 Entity Coreference

In 2010, Raghunathan et al. [87] used seven manually-defined "sieves"/passes over the corpus to conservatively resolve entity mentions. The sieves are boolean features, and examples include checking if the candidate mentions: exactly match, have matching lexical heads, compatible modifiers, have matching pronouns. This rule-based system yielded state-of-the-art results on the ACE 2004 corpus. In the same year, Haghighi and Klein [43] created a generative entity-level model that explicitly represents syntactic, semantic, and discourse constraints, and it provided the best results on the subsequent ACE 2005 corpus. In 2014, Durrett and Klein [32] created a model that used a structured conditional random field (CRF) to perform both named entity resolution and coreference resolution.

The earliest deep learning approach was by Sam Wiseman et al. [105], who trained a simple, non-linear mention-ranking model that attempts to learn distinct feature representations for anaphoricity detection and antecedent ranking. Sam Wiseman, et al. later developed an entity-level model that uses latent representations from a recurrent neural network (RNN) [104]. Clark and Manning also built both a mention-ranking model [21] and an entity-level model [22]; the former used reinforcement learning to find the optimal loss weights, and the later model merged mention-pair representations to form entity-level representations.

Lee et al. [61] were the first to demonstrate a complete end-to-end model that performs both mention detection and coreference resolution. Further, like our research, they use a relatively small number of lexical features, which we discuss in Section 4.3. Lee, et al. extended this model by using richer pre-trained word embeddings [83] and also by using a coarse-to-fine attention-mechanism to consider potential mention boundaries [62]. Zhang et al. [112] extended Lee et al.'s work by jointly optimizing both mention detection and clustering.

2.4.2 Event Coreference

In 2010, Bejan and Harabagiu [6] developed a nonparametric Bayesian model that used many lexical features, class features (e.g., part-of-speech, word classes, event classes, time, etc), WordNet features, and semantic features. In 2015, Cybulska and Vossen [25] used a decision tree with an ontology that captures varying granularity levels of locations, times and human participants, and event durations. Unfortunately, the authors used a different ECB+ testing set than that of others, along with different mentions.

In 2015, Yang, et al.'s developed a Hierarchical Distant-Dependent Chinese Restaurant Process (HDDCRP) model [108]. In short, it used logistic regression on many features, then performed clustering in a Gibbs sampling, hierarchical manner. Choubey and Huang [20] used a feed-forward neural network to iteratively group together event mentions, expanding based on the presence of their respective arguments. Kenyon-Dean, et al. [53] learn event-level cluster-influenced embeddings for mentions. Specifically, they use a feed-forward neural network to make pairwise mention predictions, but based on their custom loss function, the mentions' embeddings are updated so that ones that co-refer will have embeddings more similar to one another. More details about these systems are listed in Section 4.1.

Recently, NIST'S TAC Workshops have focused on event coreference, while making use of their own KBP 2015-2017 corpora. The KBP corpora are not publicly available, and they only concern within-doc coreference. Notable research includes novel, non-deep learning approaches: using manually-defined features and external knowledge bases (e.g., Illinois Wikification, Freebase, etc.) to construct mention vectors, then using cosine similarity for coreference resolution [81]; using Markov Logic Network with dozens of well-crafted hand-defined rules and features [67]; and using Conditional Random Fields (CRF) [66].

2.5 Event and Entity Coreference Resolution

There is little research that concerns coreference for both entities and events. Rahman and Ng [88] used event-related information to aid entity coreference. Using a mention-pair model, they created a feature to represent the semantic role of the two mentions being considered, based on their respective verb pairs. This played a small role in their overall system, as they also used external, world knowledge, and features based on the noun pairs. Humphreys, et al. [49] was the first to include entity information for event resolution, as their system constructed a discourse ontology. However, due to lack of gold truth data, they could not evaluate their system. In 2007, He's Master's Thesis [45] focused on five semantic categories within the domain of medical texts. Lee, et al. [60] used Stanford's CoreNLP [71] to resolve entities, then resolve events afterwards.

Task	Base Model	Corpus	WD-F1	CD-F1	Authors	Year
Entity	Multi-sieve*	ACE 2004	58.4	–	Raghunathan et al. [87]	2010
Entity	Graphical*	ACE 2005	53.0	–	Haghighi and Klein [43]	2010
Entity	CRF*	CoNLL-2012	61.71	–	Durrett and Klein [32]	2014
Entity	FFNN	CoNLL-2012	63.39	–	Sam Wiseman et al. [105]	2015
Entity	LSTM	CoNLL-2012	64.21	–	Sam Wiseman et al. [104]	2016
Entity	FFNN	CoNLL-2012	65.29	–	Clark and Manning [22]	2016
Entity	RL	CoNLL-2012	65.73	–	Clark and Manning [21]	2016
Entity	LSTM	CoNLL-2012	68.8	–	Lee et al. [61]	2017
Entity	LSTM	CoNLL-2012	69.2	–	Zhang et al. [112]	2018
Entity	LSTM + ELMo	CoNLL-2012	70.4	–	Lee et al. [62]	2018
Event	Cosine Sim.*	KBP 2015	39.0	–	Peng et al. [81]	2016
Event	MLN*	KBP 2015	40.08	–	Lu et al. [67]	2016
Event	CRF*	KBP 2016	33.08	–	Lu and Ng [66]	2016
Event	Bayesian*	ECB	48.6†	52.1†	Bejan and Harabagiu [6]	2010
Event	DT*	ECB+	–	60†	Cybulska and Vossen [25]	2015
Event	HDDCRP*	ECB+	66.8	58.7	Yang et al. [108]	2015
Event	FFNN	ECB+	68.9	63.6	Choubey and Huang [20]	2016
Event	FFNN	ECB+	81†	69†	Kenyon-Dean, et al. [53]	2018
Both	Sieves	ECB	–	54.2/54.8	Lee, et al. [60]	2012

Table 2.1: Overview of coreference resolution systems. WD-F1 denotes within-document CoNLL F1 score. CD-F1 denotes cross-document CoNLL F1 score. * denotes it is not a deep learning model. † denotes the scores are not comparable to the other systems that use the same corpus, either due to using a different evaluation set or metric. CRF represents a conditional random field. FFNN represents a feed-forward neural network. LSTM represents a long short-term memory network. MLN represents a Markov Logic Network. DT represents a decision tree. The Lee, et al. [60] system listed at the bottom receives 54.2 F1 for entities, and 54.8 for events.

Mentions were treated agnostic to their type (entity or event), allowing for fluid merging and mixing of clusters.

2.5.1 Summary of Coreference Systems

In summary, entity coreference has been researched more heavily than event coreference, and deep learning approaches, while on the rise, have been mostly applied to entity coreference. Further, comparing any two systems can be difficult, as it relies on testing on the same data and having used the exact same mentions (gold vs predicted). The NIST-TAC Workshops have also hosted a wave of event coreference research for their KBP 2015-2017 corpora. However, the corpora only concern within-document event coreference and are not publicly available, whereas our focus also includes the more realistic scenario of cross-document references. For a summary of the systems mentioned here, please see Table 2.1.

Chapter 3

Event Corpora

3.1 KBP 2015-2017

In the previous chapter, we highlighted a few systems that use the KBP corpora. However, the KBP corpora only concern within-document coreference and have significantly fewer documents than the ECB+ corpus. Further, the corpora are produced by NIST as part of their TAC workshops, and access to the data is limited to past workshop participants. For these reasons, we do not use KBP for our research. As a brief overview though, the KBP 2015's English training set consists of 158 documents with 6,538 event mentions that span 3,335 unique events. The English test set consists of 202 documents with 6,438 event mentions that span 4,125 unique events. Half of the corpus concerns Newswire (New York Times) articles, whereas the half is of discussion forums. KBP 2017 concerns 500 documents in total, which encompasses 8,039 unique events [38].

3.2 ECB+

We use the ECB+ corpus [24], which is the largest available dataset with annotations for event coreference. It extends the original ECB [6] corpus by exactly twice as much, and like the ECB, its documents are from the Google News Archive[1]. The ECB+ contains annotations for both within-document and cross-document, along with annotations for both entities and events. It is comprised of 43 distinct *topics* (Topics #15 and #17 do not exist), totaling 982 documents. Each topic is divided into two sub-topics. Each sub-topic is distinct, and all of its documents concern the exact same news story. However, both sub-topics for a particular topic are

[1] http://news.google.com

	Train	Dev	Test	Total
# Documents	462	73	447	982
# Sentences	7,294	649	7,867	15,810
# Mentions-1	1,938	386	2,837	5,161
# Mentions-2	142	52	240	434
# Mentions-3	18	–	25	43
# Mentions-4	6	–	7	13

Table 3.1: ECB+ Corpus statistics, where Mentions-N represents mentions that are N-tokens in length.

highly similar. For example, Topic 1a concerns Lindsay Lohan's checking into a rehab center named Betty Ford Center, which is located in Newport Beach, California. Topic 1b concerns Tara Reid's checking into a rehab center named Promises Treatment Center, which is located in Malibu, California. Each sub-topic typically contains approximately 10 documents, and each document is approximately 10 sentences in length. We maintain the same train/development/test splits as previous researchers:

- Training set: Topics 1 - 22
- Development set: Topics 23 - 25
- Testing set: Topics 26 - 45

The corpus statistics are listed in Table 3.1, where it is clear that the majority of gold mentions are one token in length (e.g, *announced*). The creators of the corpus assert that of the 15,810 sentences, they only place full faith in 1,840 of them. These 1,840 trustworthy sentences contain 5,058 event mentions:

- The Training Set has 2,117 event mentions
- The Development Set has 327 event mentions
- The Testing Set has 2,614 event mentions

These 5,508 event mentions encompass 779 unique events: 171 (22%) of which are singletons, and 608 (78%) are non-singletons. To constitute a unique event, the corresponding event mentions must all have contexts that allow one to reasonably conclude that they all occurred at the same time, location, and involved the same participants. If any of this information is missing but can be reasonably implied, the mentions are coreferent. Otherwise, they are not. As mentioned, the corpus annotates entities and events. Further, the entities are denoted with more fine-grained labels: location, time, human participant (e.g., Lindsay Lohan), or non-human participant (e.g., car).

Chapter 4

Event Coreference Resolution

We are interested in improving the performance of event coreference resolution for both within-document and cross-document environments. Past event coreference systems typically rely on using many manually-defined features, some of which are computationally expensive to compute and potentially inaccurate. In Section 4.2, we review this issue. In Section 4.3, we develop a novel mention-pair model (CCNN) which uses significantly fewer features than existing systems and frames the problem from a relational standpoint. In Section 4.4, we introduce our novel neural clustering model, which uses the CCNN's predictions to holistically cluster event mentions. Our CCNN mention-pair model and Neural Clustering model combine to deliver state-of-the-art results in event coreference and runs in just a few minutes on a single GPU (e.g., NVIDIA Titan X).

4.1 Related Work

The majority of our research uses the ECB+ corpus [24], which we further describe in Section 3.2. This rich corpus provides annotations for both entities and events, yet, as mentioned, most research focuses on using either events *or* entities, not both. There are three papers which research cross-document event coreference and make use of the ECB+ corpus; naturally, these three systems are the most relevant and comparable to our research.

- The Hierarchical Distance-dependent Chinese Restaurant Process (HDDCRP) model by Yang et al. [108]
- Iteratively-Unfolding approach by Choubey and Huang [20]

- Resolving Event Coreference with Supervised Representation Learning and Clustering-Oriented Regularization (CORE) [53]

4.1.1 HDDCRP Model

Yang et al.'s HDDCRP model [108] takes a mention-pair approach: they use logistic regression to train feature parameters θ (see Section 4.2) across all pairs of mentions. This regression model serves as the similarity function in Equation 4.1. Then, in a Chinese-restaurant-process fashion, they probabilistically link together mentions based purely on the scores provided by this similarity function. That is, the value of $f(m_i, m_j)$ is directly correlated with the probability of (m_i, m_j) being chosen as a linked pair. Then, identical to Bengtson's and Roth's work [10], the HDDCRP model forms clusters by tracing through all linked pairs. All mentions that are reachable by a continuous path become assigned the same cluster. This hinges on the transitive property of coreference. For example, if $(m_1, m_3), (m_3, m_5)$ and (m_5, m_6) are each individually linked via the scoring function, then a cluster $C_i = \{m_1, m_3, m_5, m_6\}$ is formed, even though (m_1, m_5) or (m_3, m_6) may have had very low similarity scores. We aim to improve this shortcoming, as detailed in Section [TODO fix refs]

$$f_\theta(x_i, x_j) \propto \exp\{\theta^T \psi(m_i, m_j)\} \tag{4.1}$$

The hierarchical aspect of HDDCRP comes from their performing coreference on a within-document basis first, then using these formed clusters as the starting clusters for cross-document coreference. This approach is highly effective, in part because the cross-document scenario has drastically more candidate pairs than the within-doc; thus, positive co-referent pairs comprise a lower percentage of the total possible pairs, yielding a "needle-in-the-haystack" problem. If we were to perform coreference on a cross-document scenario without any prior within-doc clustering, the task would be more difficult.

4.1.2 Neural Iteratively-Unfolding Model

Choubey and Huang [20] introduced the first neural model for event coreference. Their system is also a mention-pair model, and coreferences are predicted by a feed-forward network. The novelty of their system is that they (1) supplemented every event mention with SRL information (argument time and location); and (2) performed additional cluster merging based on two hard-threshold rules — merge clusters if they contain any pair of mentions that either share a governing entity or have contexts that are similar beyond a particular threshold.

The authors assert that when using the ECB+ corpus, within-doc coreference did not benefit from using mention context, which is an important finding. However, similar to the weakness of the HDDCRP model, they merge clusters which contain *any* mention-pair whose predicted score is below a given threshold, independent of mentions' relation to the cluster at large.

4.1.3 CORE

Resolving Event Coreference with Supervised Representation Learning and Clustering-Oriented Regularization (CORE) [53] also takes a mention-pair approach, whereby every pair of mentions is fed into a feed-forward neural network; the novelty is that the middle hidden-layer is explicitly adjusted according to the *cluster* to which each mention belongs. That is, each unique cluster (coreference chain) will cause all its mentions to have hidden embeddings that are similar to one another, while mentions in different clusters will become dissimilar. Using the inferred hidden embeddings, the system then performs agglomerative clustering.

This approach is possible because the training set of the corpus provides gold annotations for which mentions should all co-refer. Thus, input mentions' embeddings are mapped to hidden representations that adhere to this similarity-property. However, at test time, the system obviously cannot use gold cluster information, as that is the task at hand. Testing performance thus hinges on (1) having mentions that are similar to the ones seen during training; and, (2) having a similar number of unique clusters as in train; (3) having similar prediction scores as that from the development set. The latter points are necessary because any drastic change will affect inferred hidden embeddings and their respective distance scores.

4.1.4 Convolutional Neural Networks (CNN)

Since we want our model to automatically discover the most useful mention-pair features, or subsets thereof, we turn to CNNs, which are known for the ability to effectively and robustly use sub-regions of inputs (see Section 2.2.2). Convolutional Neural Networks (CNNs) are one of the most common neural network architectures, and they have provided compelling results for many NLP tasks, including sentence classification [54], machine translation [37], dependency parsing [109], mention detection [17, 78, 79], and relation classification [29, 110] just to name a few.

	HDDCRP [108]	Choubey [20]	CORE [53]
Mention Features			
String Match	✓		
Part-of-Speech (POS)	✓	✓	
Token Similarity			✓
Word Embeddings	✓		
Lemma Embeddings		✓	✓
WordNet	✓		
Position			✓
Cluster Comparison			✓
Participant Features			
SRL-based	✓	✓	
Time	✓	✓	
Location	✓	✓	
Context Features			
Word Similarity	✓	✓	✓
POS	✓		
Document Embeddings			✓

Table 4.1: The categories of features most commonly used by coreference systems, and we explicitly denote the usage by the three models most similar to our system.

4.2 Common Event Coreference Features

Coreference resolution systems typically use a plethora of features — usually ten to fifty manually-created ones (e.g., boolean value representing if the two candidate mentions both contain context words that share the same part-of-speech). Systems generally create features based on the mentions themselves, their context, and occasionally the participants/arguments identified from SRL systems. In Table 4.1, we summarize the features used in the three most relevant event coreference systems. Note: each category listed could encompass several smaller, fine-grained features. For example, the CORE paper's features include checking if the two candidate mentions: (1) have the same first token; (2) the same last token; (3) have all tokens the same; (4) have contexts that are identical in their 1st two words, last two words, all 4 words, etc. Yet, we group all of these features together and merely denote it as one ✓ for "token similarity" and one ✓ for context "word similarity."

4.3 Conjoined Convolutional Neural Network

4.3.1 Motivation

As discussed in Section 1.3.2, the common paradigms for coreference resolution include mention-ranking, mention-pair, and entity/event-level. While not a single one of these is undeniably superior to the rest, we were

motivated to develop a mention-pair model for the following reasons: entity/event-level models operate by building a single representation for each underlying entity/event, which is predicated upon knowing how many unique entities/events exist within each document(s) — this effectively encompasses both tasks of scoring mention coreference and clustering. While this may seem like an attractive property, it can be difficult [104] to (1) estimate a priori how many unique events exist; and (2) devise a neural, event-level representation that is robust to adjustments as events are added and removed from each cluster.

Mention-ranking models circumvent these difficulties by distinctly scoring mentions separately from any subsequent clustering. However, mention-ranking models can be effectively reduced to mention-pair models, with the added complexity of needing to explicitly model the likelihood that any given mention does not co-refer with any other mention (i.e., is a singleton). For example, a mention-ranking model decides the likelihood that a mention is a singleton or not, and in the latter case, it links the mention m_j to the mention m_i that appears earlier in the text which has the best coreference score — this part is identical to the mention-pair framework of looking at all possible pairs and picking the mention m_i that matches best with the given mention m_j. If one had an abundance of annotated data, we would argue that event-level or mention-ranking models would be a strong choice; however, given the relatively limited amount of annotated coreference data that exists, we are interested in developing a mention-pair model. Further beneficial, mention-pair models tend to be rather straight-forward and with reasonable computational complexity. We want our model to:

- rely on few hand-crafted features (e.g., WordNet or FrameNet), ideally by merely providing our system the raw text and let it learn how to represent the mentions
- model the *relationship* between two given mentions, as opposed to explicitly defining relational features (e.g., Jaccard Similarity between bag-of-word context windows)
- be robust to out-of-vocabulary items during test time

A Conjoined Convolutional Neural Network aligns with our interests.

4.3.2 Background

Conjoined Networks (a.k.a. Siamese Networks) are two identical neural networks that are joined together to produce a similarity score — representing the likelihood that the two inputs are the same. Each neural network accepts distinct inputs. The networks are said to be *conjoined* because they share the same weights, have a single loss function, and thus work together as one network that learns a task. Conjoined Networks were first

introduced by Bromly and LeCun [14] for the task of determining if two signatures were from the same person or not. The benefits of tying the weights are that: (1) it ensures that similar inputs will be mapped appropriately, otherwise, they could be mapped to hidden representations that are dissimilar from their input representations; and (2) it forces the network to be symmetric. This is critical, as the network's similarity function should be independent of the ordering of its input pair. If we abstractly represent the Conjoined Network as a function, then:

$CN(f_i, f_j) \equiv CN(f_j, f_i)$

Note, the exact architecture of neural networks that we choose to conjoin could be of any type — feed-forward, recurrent, convolutional, etc. We now discuss our choice: convolutional networks.

4.3.3 Model

Our model is a Conjoined Convolution Neural Network (CCNN). Each input pair corresponds to a pair of event mentions, and the output corresponds to how similar the mentions are to each other. Using conjoined networks allows our model to learn relationships (i.e., similarities) between mentions. Since our model does not explicitly rely on learning actual input values, but rather the similarity between the abstract representations of inputs, our system is theoretically robust to out-of-vocabulary (OOV) event mentions at test time. Related, Conjoined Neural Networks have been shown to perform well in low-resource situations [41]. Using convolution as our choice of conjoined network architecture allows our model to appropriately weight sub-regions of input embedding features and the relationships thereof.

Input Features

Our CCNN needs each mention to be represented by its own feature embedding, and we deliberately use no hand-defined relational features that are common in other coreference systems (e.g., binarized same-gender or same-speaker, Jaccard similarity of mentions' context, cosine similarity of context word embeddings). First, we pre-process our corpus by using Stanford CoreNLP Toolkit [70] to extract the part-of-speech tags and lemma for every word, along with the dependency parse of every sentence. We also import GloVe word embeddings [82] (the 42-billon token crawl). Having done so, we can extract the following five features for every pair of mentions:

- **Part-of-Speech:** We experimented with (1) representing our corpus by replacing every word by its corresponding POS tags, then running an LSTM over the corpus to construct token-based POS embeddings;

and (2) representing each POS tag as a random vector and using them as 1-hot embedding lookups. The latter worked better (likely due to the variance of context, which we address later), so all POS-feature results use such.

- **Lemmatization**: We use the pre-trained GloVe word embedding that corresponds to the lemma of every word token. For example, if the word is *running*, the lemma is *run*, and we use GloVe's embedding for *run*.

- **Dependency Lemma:** Similar to the above feature, but instead of using the lemma of the mention's word, we use the GloVe embeddings of the lemma of the dependency parent and dependency children. In case of having multiple dependency parents or children, we experimented with: (1) using an average of all; (2) using the word that is the longest; (3) using the word that belongs to any other entity or event mention. The average performed best, so we report results for such.

- **Character Embeddings:** We experimented with: (1) pre-training GloVe on a character-tokenized representation of a large Google News corpus; (2) using randomly initialized character embeddings. The former performed better, so we report its results.

- **Word Embeddings:** pre-trained GloVe word embeddings (we tried both the 42-billion and 840-billion crawls, and the former had better results more often, but it was essentially negligible).

We account for mentions' having varying token lengths by summing their tokens in place, thus representing each mention as a fixed-length vector. For example, if the mention were *Barack Obama*, we would sum the embeddings of *Barack* and *Obama*. Averaging gave worse results, possibly because it loses more information than summing.

Architecture

The full architecture is shown in Figure 4.1. We define the full embedding for a given token t as $t_{emb} = t_{f_1} \oplus t_{f_2} \oplus \ldots \oplus t_{f_n}$, where $\oplus$ represents vector concatenation and t_{f_i} represents a specific input feature vector. We use the context of mention m by including the N words before and after m. Thus, our input for mention m is a matrix M, where each row corresponds to a token t_{emb} of length d, and, a la Kim [54], we zero-pad unfilled context windows.

Let **M** represent the full matrix corresponding to mention m. If we are using lemma embeddings (with 300 dimension vectors) and character embeddings (with 400 dimension vectors), along with a context window size

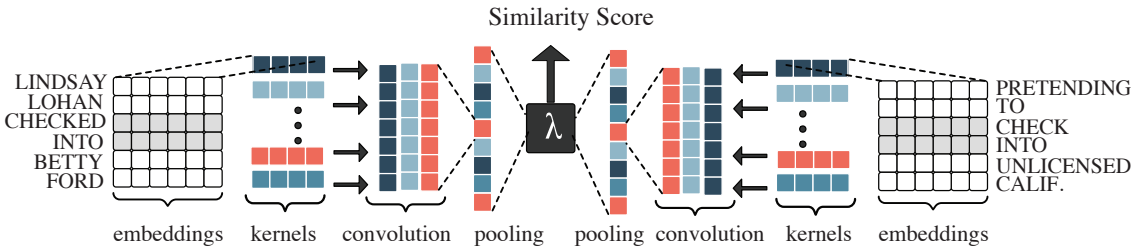


Figure 4.1: The Conjoined Convolutional Neural Network's (CCNN's) Architecture. A pair of event mentions is input into the network, and the output is the likelihood of the two mentions being co-referent.

of 3 words before and after the mention, our matrix M will be of size 7x700. Equivalently, $\mathbf{M} \in \mathbb{R}^{(2N+1)\times d}$. Further, let $\mathbf{M}_{(i,j),(k:l)}$ represent the sub-matrix of M from (i,j) to (k,l). We define a kernel with dimensions (h,w), where $h < (2N+1)$ and $w < d$. This allows the kernel to operate on sub-sections of the embeddings. The kernel has an associated weight matrix $\mathbf{w} \in \mathbb{R}^{h\times w}$. Starting at a given index (i,j) within mention matrix $\mathbf{M}$, a feature c_i is defined as:

$$c_i = f(\mathbf{w}^T \mathbf{M}_{(i:i+h-1),(j:j+w-1)} + b) \tag{4.2}$$

where $b \in \mathbb{R}$ is an added bias term. The kernel iteratively slides over every possible sub-section of mention matrix $\mathbf{M}$ (stride of 1), yielding a feature map $\mathbf{c} \in \mathbb{R}^{(2N-h)\times(d-w-1)}$. The Lambda function calculates the L^2 distance (Equation 4.3) of each half's univariate vector and emits a prediction of the two mentions being co-referent or not.

$$\begin{aligned} L^2(x,y) &= \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2 + \ldots + (x_n-y_n)^2} \\ &= \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \end{aligned} \tag{4.3}$$

Loss / Optimization

Our model's goal is to maximize discriminability of different event mentions, while enforcing features to be as similar as possible when they are of the same event. Thus, we define our output prediction to be a similarity score s, where $s = 0$ when the two input mentions should co-refer, and $s = 1$ when they should not. Unlike many neural networks that aim to predict a discrete category, we are predicting a distance score. We use

contrastive loss, a distance-based loss function shown in Equation 4.4 [65].

$$L(x_1, x_2, y) = \frac{1}{2N}\sum_{n=1}^{N}[(y)d^2 + (1-y) * (max(m-d, 0))^2] \tag{4.4}$$
$$\text{where } d = L^2(\text{CCNN}(x_1), \text{CCNN}(x_2)) \text{ and } m = \text{the ceiling distance score}$$

We use the *Adam* algorithm [55] for optimization.

4.3.4 Experiments

Corpora

We use ECB+, as it is the premier event coreference corpus and contains both within-doc and cross-doc annotations. We adhere to the standard data splits: development set is topics 23-25, and the test set is topics 26-45. Traditionally, topics 1-22 are used as training. Since our NC model relies on our CCNN's predictions, we remove topics 19-22 from the training set and instead use them as development sets for our NC models. **Note:** the KBP corpora contain only within-doc annotations and are not publicly available.

Mention Detection

Determining which words constitute a mention is the first sub-task of coreference resolution and is often a separate line of research called *event detection*. To fairly evaluate our coreference system against the other existing cross-document event coreference systems, we extract precisely the same mentions as they — otherwise, any performance gains may be attributed to simply having more accurately identified mentions. Thus, we use the same mentions as Yang et al. [108] and Choubey and Huang [20]; mentions were produced from training a semi-Markov CRF [92] on the ECB+ training set, with a loss function defined by [108].

Within-Document and Cross-Document

We train and evaluate our CCNN on both within-document and cross-document scenarios. The former is straightforward: only mentions contain in the same document will be considered. For the cross-document scenario, we follow suit with most other research by considering mention pairs that are explicitly in different documents. Thus, these two scenarios concern disjoint pairs of mentions, but together, they concern every possible pair of mentions within the corpus.

Hyper-parameters

The CCNN's hyper-parameters used for the test set were selected based on which values yielded the best performance on the development set. Specifically, we used 2 layers of convolution, 10 epochs of training, a context window size of 0 (discussed in Section 4.4.8), 5 negative training examples per positive training example, a batch size of 64, a dropout rate of 0.4, and 32 convolution filters. The training data was down-sampled to a 5:1 ratio because of the strong class imbalance (most input pairs are not co-referent).

Ensemble

There are stochastic elements to our CCNN, such as the initial, random weights and our shuffling the training data between each epoch. This causes slight variations in our results. In an attempt to minimize these variations, we perform an ensemble approach, whereby our final event-pair predictions are based on combining the predictions (on a per-pair basis) across 50 individual runs.

Evaluation Metrics

Our CCNN makes pairwise, continuous-valued predictions, which will serve as input to our NC. Thus, any evaluation here can be viewed as intermediate results en route to our final clustering performance. What ultimately matters is how well our CCNN orders/ranks its predicted pairs; the better the ranked list, the better the NC will perform. Thus, to measure performance, we can set a threshold such that all items with a score less than the threshold are considered to be predicted as positive pairs that should co-refer, and items above the threshold are negative pairs that should not co-refer. In transforming our predictions to boolean-valued predictions, we can measure our overall accuracy (e.g. "how many pairs are correct?"). Yet, this is not too meaningful or fine-grained because most pairs should not co-refer, so one can have a misleadingly high accuracy score by merely reporting no pairs should co-refer. Alternatively, let TP represent our returned true positives, FN represent our returned false negatives, and FP represent our returned false positives. Then, *recall*, *precision*, and *F1* are defined as:

$$recall = \frac{TP}{TP + FN} \quad \text{and} \quad precision = \frac{TP}{TP + FP} \quad \text{and} \quad F1 = \frac{2 * precision * recall}{precision + recall}$$

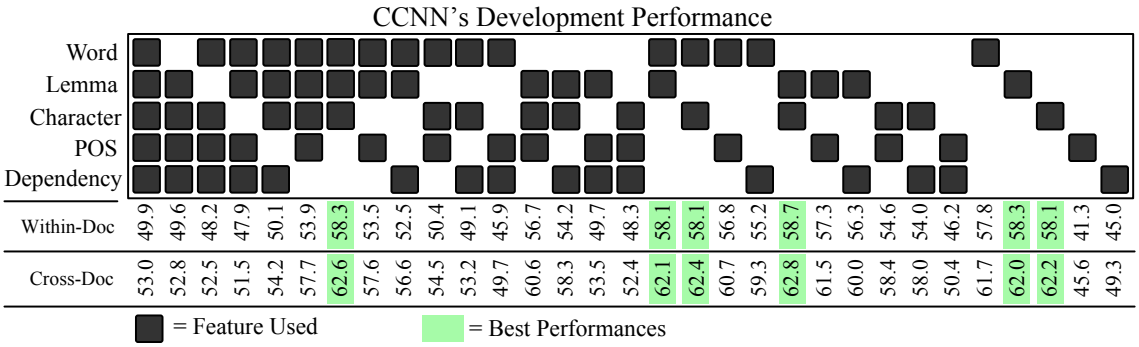


Figure 4.2: The CCNN's mention-pair performance, using all combinations of features. Scores reported are F1.

4.3.5 Results

Using gold truth clusters, we can measure the performance of mention-pair models by their accuracy in predicting if two given mentions should co-refer or not. Figure 4.2 shows our CCNN's results with every possible combination of features. Notably, we see that using *Lemma Embeddings* and *Character Embeddings* together yield the highest results. Considering *SameLemma* is always a depressingly strong baseline, it should be no surprise that lemma *embeddings* are useful. When used in conjunction with character embeddings, we believe they serve as complementary features — the former provides semantic information while the latter provides syntactic information. Specifically, the *character embeddings* feature is similar to using String Edit Distance as a feature — merely calculating how many character-changes one would need to make to convert one mention to another. To see why this is useful, imagine we had two mentions: "5 a.m. earthquake" and "earthquake." While the lemma embedding for 5 a.m. will completely alter the mention's overall lemma embedding, the character embedding for each mention will be very similar, thus encouraging our CCNN to give them a high similarity score. In theory, a model should be able to learn which of its features are unhelpful and appropriately give them little-to-no weight. However, due to our corpus being relatively small, and due to the high variability in golden coreference chains, we assert that: (1) it is hard for any coreference model to learn appropriate weights for many features; and (2) systems should start using as few features as possible.

The cross-document scenario involves many more mention-pair predictions than the within-document scenario (8,939 vs 775, for the small development set), and thus many more candidates and lower % of golden coreferent pairs. Thus, one may expect a lower cross-document performance. However, due to the larger training size, and due to the more parallel lexical representation in the cross-document scenario, our cross-document mention-pair performance exceeds within-document.

In Table 4.2, we report the CCNN's best performing results (Lemma + Character Embeddings), along with

	Precision	Recall	F1
	Within-Document		
SameLemma$_{any}$	53.9	48.0	50.8
SameLemma$_{all}$	50.3	46.3	48.2
LibSVM	51.2	52.0	51.6 (0.01)
FFNN	50.3	59.8	54.6 (0.5)
CCNN	51.5	68.2	**58.7 (0.8)**
	Cross-Document		
SameLemma$_{any}$	55.6	54.1	54.8
SameLemma$_{all}$	53.1	51.0	52.0
LibSVM	58.6	59.1	58.8 (0.02)
FFNN	55.3	62.0	58.5 (0.6)
CCNN	55.8	71.2	**62.8 (0.6)**

Table 4.2: Models' mention-pair (not clustering) performance on the development set. This can be viewed as intermediate results, as these mention-pair predictions are used by our clustering algorithm. Each score is the average of 50 runs, with standard dev. denoted in ( ).

Context Size	Within-Document	Cross-Document
0	58.7	62.8
1	58.4	62.1
3	57.9	60.7
5	57.1	60.5

Table 4.3: F1 performance on the ECB+ Development Set while varying CCNN's context-window size (how many words on each side of the event mention to include in CCNN's input.)

how it compares against other strong classifiers: *SameLemma*$_{any}$ simply denotes any two mentions as being co-referent if *any* of their words have the same lemma, which has proven to be a historically strong baseline. Similarly, *SameLemma*$_{all}$ denotes any two mentions as being co-referent if *all* of their words have the same lemma. LibSVM and a Feed-Forward Neural Net (FFNN). The FFNN uses 2 hidden layers of size 300 and 100, ReLU activation, and Adam optimization. These baselines receive the same input embeddings as our CCNN. In addition, we measure performance as we vary the training size, as shown in Figure 4.3. The results suggest that more training data is needed, furthering our assertion that it is imperative to not include too many features in a system, as doing so would add additional complexity to the already-limited data.

In an attempt to measure the effect of context, we experimented with providing our CCNN different context-window sizes, while fixing the features to be the optimal lemma + character embeddings. As shown in Table 4.3, results on the ECB+ Development Set indicate that increasing the amount of context monotonically decreases performance. This may seem counter-intuitive, as context provides more information; yet, our model is unable to soundly use this information. This agrees with the results of Choubey [20], whereby context, especially for the within-document scenario, did not improve results. We address this further in Chapter 5.

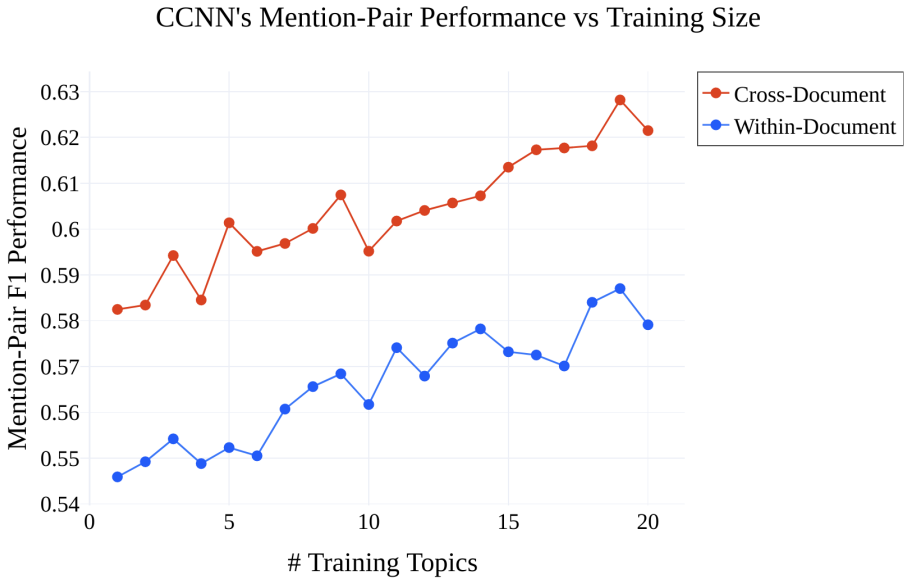


Figure 4.3: The effect the size of training has on the ECB+ development set performance.

4.3.6 Error Analysis

First, it is important to realize that what matters most is the ranked ordering that our CCNN provides, not the actual prediction values. Since our Neural Clustering will take the CCNN continuous-valued predictions as inputs for its clustering, not boolean, hard-valued coreference decisions (e.g, 0 or 1), it has the ability to learn how to use these predictions and what range of values are meaningful. Thus, we should evaluate the *ranked order* of the CCNN prediction list, while using any threshold that yields the highest F1 score. When doing so for the cross-document scenario of 8,939 individual mention pairs, the optimal threshold yields 86 false positives and 569 false negatives. Of these, I manually expected all false positives and 100 of the false negatives, while categorizing the types of errors we made. Examples of the false positives and false negatives are shown in Tables 4.4 and 4.5, respectively. Note: while any given coreference-pair error could often times fit within multiple categories, I counted it only towards the category I deemed most appropriate.

Among the false positives, the *Context-Dependent* category concerns cases where our system wrongly thought two mentions co-refer because the mentions had identical (or near-identical) lexical representations (e.g., announced and announced). Clearly, factoring in the context, in particular, the entities involved, would have helped. Likewise, the *Similar Meanings* category is when our system wrongly linked mentions due to their having similar lemma embeddings, and including context could have helped. *Wide-Reading* is when an event is roughly subsumed by the other event, and because they are very similar, our system wrongly linked them

False Positives	
Context-Dependent (30%)	
Example 1	The 55-year-old Scottish actor will replace Matt Smith, who **announced** in June that he was leaving the sci-fi show later this year.
	Peter Capaldi has been **announced** as the new Doctor Who, the 12th actor to take up the coveted TV role.
Similar Meanings (38%)	
Example 2	Frederick C. Larue, a top Nixon campaign official who **passed** money from a secret White House fund, **died** Saturday at a hotel in Biloxi, Miss.
Wide-Reading (14%)	
Example 3	Peyton manning helped inspire the Indianapolis Colts to their eighth straight **win** as they **overcame** Jacksonville this season.
Unclear (13%)	
Example 4	Microsoft today issued an emergency update to **plug** a critical security hole present in all version of its browser, a flaw hackers have **used** to steal data from millions of Windows users.
Syntax (3%)	
Example 5	Creighton **defeats** Drake 65-53 in MVC tournament.
	In Saturday's semi-finals, Creighton will play no. 5 seed Indiana state, which **defeated** Evansville 51-50 on Friday.
Too Difficult for Me (2%)	
Example 6	Submarine cable **problem** disrupts telecom services in Alexandria .
	Vodafone has been affected by a **damage** in one of the fiber cables going from the Ramsis Communication Center all the way to Sadat City.

Table 4.4: Examples of CCNN's False Positives from the ECB+ Development Set, grouped by categories of errors.

False Negatives	
Semantics (42%)	
Example 1	Hansbrough scored 20 points Thursday night, **breaking** North Carolina's career scoring record, and the tar heels beat visiting Evansville, 91-73 .
	Hansbrough **sets** scoring record in victory .
Unclear (20%)	
Example 2	Hewlett-Packard's **purchase** of electronic data systems could mean tougher competition for IBM and its 10,500 triangle employees.
	The all-cash **deal**, announced Tuesday, represents HP's biggest gamble under the leadership of Mark Hurd.
Colloquial Variations (16%)	
Example 3	Industry experts told The Times that two sub-sea cables **went down** just off Alexandra, causing mass disruption.
	Millions of people across the Middle East and Asia have **lost** access to the Internet after two undersea cables in the Mediterranean suffered severe damage.
Longer Names (14%)	
Example 4	An **earthquake** with a preliminary magnitude of 4.6 was recorded in the North Bay this morning, according to the U.S. Geological Survey.
	A **4.6-magnitude earthquake** was recorded near Healdsburg .
Pronouns (8%)	
Example 5	President Obama announces **nominee** for surgeon general.
	Today, President Barack Obama announced his intent to **nominate** Regina M. Benjamin as surgeon general, department of health and human services.

Table 4.5: Examples of CCNN's False Negatives from the ECB+ Development Set, grouped by categories of errors.

together as being the same. *Unclear* is when I am unsure why our system linked together two mentions; the combination of lemma embeddings and character embeddings are unfortunately oddly close. In the provided example, the event mentions "plug" and "used" both have exactly four characters, and since that number is low, and both words share a "u," our averaged character embeddings are similar to each other. *Syntax* is when the event mentions' words are lexically similar to each other, causing their character embeddings to be similar. The final category is when the context alone is insufficient for me to determine if the mentions should co-refer, so it is no surprise that our system incorrectly predicts them.

Concerning the false negatives, the *semantics* category is again the most common error. Here, we miss mention pairs that have the same meaning, as our lemma and character embeddings did not capture such — and using context could have helped us. In contrast, our *semantic* false positive errors were due to our linking two mentions thought to have the same meaning (based on their embeddings), yet again, factoring in context could have helped. The *unclear* false negatives are cases when I am unsure why our system failed to link the mentions — sometimes the lemma embeddings are too disparate, sometimes our character embeddings are so. *Colloquial Variations* is when the mentions are understandably difficult to handle, as the wording

would require external, human-level knowledge. These, along with the *longer names* and *pronouns* categories, are highly difficult. For example, slang phrases like "stepped into the role" and "hired" are co-referent, and capturing such seems non-intuitive, as it might require access to external knowledge bases (e.g., paraphrases).

4.4 Neural Clustering

4.4.1 Motivation

It is common practice for coreference systems determine their final clusters via agglomerative clustering [58, 108]. Agglomerative Clustering first assigns each mention to its own singleton cluster then repeatedly merges the two distinct clusters which contain the shortest-distance mention pairs. Although this is a strong baseline, there are three main weaknesses:

1. One must define a stopping threshold α.
2. Any given α hinges on the data being uniform across documents. In reality, distances between mention-pairs could vary significantly between documents and topics.
3. Each cluster merge is based solely on two individual mentions, yet these mentions may not be representative of the cluster at large.

HDDCRP and Iterative-Folding (Choubey) both contain issue #3, as detailed in Sections 4.1.1 and 4.1.2, respectively.

4.4.2 Model

We use the strengths of agglomerative clustering while replacing its shortcomings. Instead of predicting individual mention-pair merges, we learn a function $f(C_x, C_y)$ that predicts the likelihood of merging *clusters*. Let $d(m_i, m_j)$ be the mention-pair distance predicted by our CCNN model, where $m_i \in C_x$, and $m_j \in C_y$. Function $f(C_x, C_y)$ is based on four simple features:

- min-pair distance: $\min_{m_i,m_j} d(m_i, m_j)$
- avg-pair distance: $\frac{\sum_{m_i,m_j} d(m_i,m_j)}{\|C_x\|\|C_y\|}$
- max-pair distance: $\max_{m_i,m_j} d(m_i, m_j)$

- candidate cluster size: $\frac{\|C_x\|+\|C_y\|}{\sum_z \|C_z\|}$

The first three features serve to better represent the cluster at large (issue #3 from above). For example, when evaluating a cluster C_1, it may have the same minimum mention-pair distance score with two candidate clusters C_2 and C_3. Yet, the average and maximum distance scores reveal which clusters have more similar mentions. The *candidate cluster size* feature represents the size percentage of our considered merge, relative to all mentions. This helps prevent clusters from growing too large, addressing issue #1 from above.

4.4.3 Architecture

We define f as a feed-forward neural network which predicts a softmax probability of a positive cluster merge. We used 1 hidden layer of 25 units, ReLU activation without dropout, a learning rate of 0.001, and Adam optimization. Our loss function was weighted binary cross-entropy, to account for the class imbalance situation (most pairs of clusters should not be merged together).

4.4.4 Training

At test time, our system incrementally merges clusters, starting with each cluster having just one mention (in the within-document scenario). Since we will encounter decisions to merge clusters of varying sizes, we need to train our clustering model on such scenarios. Within the gold training data, there is obviously no single canonical ordering to which co-referent mentions formed a cluster; thus, we need to generate synthetic cluster-merge data to represent positive and negative examples of when clusters of varying sizes should be merged. Specifically, for training, we generate a positive example by randomly sampling a golden cluster and splitting the cluster into two random subsets (see Algorithm 1). The above four features are calculated for these two subsets of clusters, and the target output is a positive case. Likewise, we generate negative examples by sampling random subsets from disjoint golden clusters.

4.4.5 Evaluation

At test time, we use Neural Cluster to evaluate every possible (C_x, C_y) cluster pair in an easy-first manner. That is, at each iteration, we merge the (C_x, C_y) pair that yielded the highest likelihood of a positive merge. Then, we re-evaluate all cluster-pairs and repeat until the model no longer predicts a merge. Thus, unlike the aforementioned models, we do not require additional stopping parameters.

Result: constructs training of size N
training = {};
while $|training| < N$ **do**
    pick two random golden clusters C_1, C_2;
    let $C_j, C_k = get\_subset(C_1)$;
    let $C_l, C_m = get\_subset(C_2)$;
    training.add((C_j,C_k) features, True));
    training.add((C_j,C_l) features, False));
Function *get_subset*(cluster C_i)
    let $C_x, C_y = \{\}$;
    pick a random size $S \Leftarrow |C_i|$;
    while $|C_x| + |C_y| < S$ **do**
        randomly pick $m \in C_i$ and $m \notin \{C_x, C_y\}$;
        place m in random cluster C_x or C_y;
    return C_x and C_y;

Algorithm 1: Construct NC Training Data

Within-Document

The within-document scenario is straight-forward: the NC performs clustering with CCNN's within-document pairwise predictions.

Cross-Document

Cross-document resolution is a superset of the within-document task; it uses all coreference chains, regardless if mentions in a cluster were originally from the same document or not. Our cross-document and within-document systems are identical, except: (1) we train a separate CCNN only on mention-pairs which are from different documents; (2) instead of initializing our clustering with all singleton clusters, we use our within-document NC predictions as starting clusters; (3) at each iteration, we only consider merging clusters (C_x, C_y) if C_x and C_y contain mentions from disjoint sets of documents. Our cross-document NC only uses cross-document mention pairs distances for its decisions. Thus, cross-document merging will never merge two within-document clusters from the same document.

Metrics

Instead of evaluating mention-pairs, we are now interested in evaluating clusters. The three most commonly used metrics are *MUC*, B^3, and $CEAF_e$. Of these, there is no canonical best, as they all address different aspects [74, 101]:

MUC [102] measures the accuracy of the coreference links, as it counts the minimum number of edge-insertions/deletions necessary to obtain the gold clustering from the predicted clustering. This is akin to String

Edit Distance, but for links. Notably, since it is link-based, MUC does not factor in singleton mentions which do not cluster with other mentions.

B^3 [4] measures the proportion of overlap between the predicted and golden coreference chains, yielding precision and recall scores – ultimately B^3 returns the average over the scores across all mentions. While this accounts for singletons, unlike MUC, it also potentially uses events/entities of the same coreference chain more than once.

CEAF$_e$ [68] is an event-/entity- level metric that finds an optimal alignment between the predicted chains and gold coreference chains by maximizing a similarity objective, which in turn is used to calculate precision and recall.

Since no single metric is best, the *CoNLL F1* score was created, which is the average F1 score from each of the aforementioned three metrics. This is the standard metric used by coreference systems, so we follow suit. Further, we use the official scorer script (v8.01) [84] that was provided at past CoNLL workshops.

4.4.6 Results

Having run our full system, CCNN + NC, we can once again evaluate our clustering performance on the development set to see the effects of different features (see Figure 4.4. Since the NC takes as input the CCNN's predictions, and its performance hinges on how well the CCNN performs, we expect to see no significant differences between CCNN's results and NC's in terms of the relative impact of particular feature combinations – any difference in order is only due to variance between runs. Not unsurprisingly, we see that using lemma embeddings + character embeddings yields the best performance, so we use this feature combination on the test set for final evaluation. In Table 4.6, we report our performance on the ECB+ test set compared to other systems. Notably, our CCNN performs better than other baselines, and NC always performs better than the standard agglomerative clustering approach. In general, cross-document CoNLL F1 scores will be naturally lower than within-document, unlike when evaluating on a pairwise-mention basis. This is simply because there is more room for error on a cluster-wide basis, evident by cross-document MUC scores (a mention-based metric) being higher than within-document MUC scores, but B^3 and CEAF (cluster-based) having lower scores than within-document.

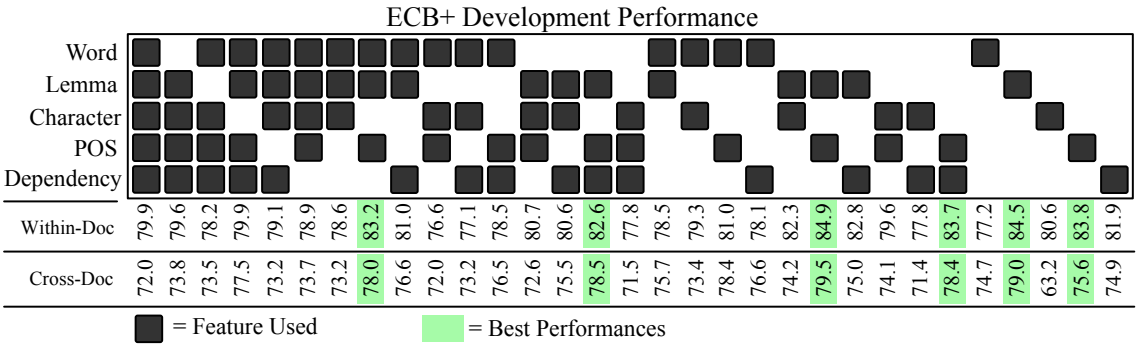


Figure 4.4: The clustering performance of our flagship CCNN + Neural Clustering system, using all combinations of features. Scores reported are CoNLL F1.

	Within-Document				Cross-Document			
	MUC	B^3	CEAF	CoNLL F1	MUC	B^3	CEAF	CoNLL F1
SameLemma_{any}	40.4	66.4	66.2	57.7	66.7	51.4	46.2	54.8
HDDCRP [108]	53.4	75.4	71.7	66.8	73.1	53.5	49.5	58.7
Choubey [20]	62.6	72.4	71.8	68.9	73.4	61.0	56.5	63.6
FFNN+AGG	61.6	73.6	69.1	68.1 (0.14)	74.8	55.3	60.2	63.4 (0.21)
FFNN+NC	62.5	73.2	70.8	68.8 (0.17)	76.1	56.0	60.4	64.2 (0.18)
CCNN+AGG	65.2	74.2	69.0	69.5 (0.16)	75.8	55.8	62.7	64.8 (0.21)
CCNN+NC	67.3	73.3	69.6	70.1 (0.20)	77.2	56.3	62.0	65.2 (0.22)
CCNN+NC (ensemble)	**67.7**	**73.6**	**69.8**	**70.4 (0.13)**	**78.1**	**56.6**	**62.1**	**65.6 (0.17)**

Table 4.6: Coreference Systems' clustering performance on the ECB+ test set, using the predicted mentions and testing procedure from Choubey and Huang [20]. Our CCNN models use only the Lemma + Character Embedding features. FFNN denotes a Feed-Forward Neural Network Mention-Pair model. AGG denotes Agglomerative Clustering. Our models' scores represent the average from 50 runs, with standard deviation denoted by ( ).

4.4.7 Comparison to Other Systems

Using the same mentions as Choubey and Huang [20], our flagship CCNN+NC system outperforms all models, despite using few features (see Table 4.6). In particular, our clusters tend not to amass too many mentions, whereas other systems may merge two disjoint, spurious clusters due to having just a few shared related mentions. Regardless of the model, cross-document coreference clustering involves many more mentions than within-doc, naturally yielding a larger margin for clustering errors and thus lower CoNLL F1 scores.

4.4.8 Conclusions

We have shown that it is possible to yield state-of-the-art results on the ECB+ corpus while using just a few features. We suspect this is exacerbated by the relatively small amount of training data that exists for coreference. Further, we identified weaknesses in current agglomerative-based clustering approaches, which we improve upon with our more holistic, neural-based clustering (NC). Despite these improvements, it is clear from our system's errors that using context can improve performance, which is the focus of our next chapter.

Chapter 5

Joint Entity and Event Coreference

5.1 Motivation

Our primary focus is to event resolution, and the last chapter posits that results can be improved by using context better, including entity information. Figure 5.1 illustrates two examples from the ECB+ corpus: (a) sentences with event mentions that are difficult for coreference resolution, but their respective, associated entities are relatively easy to resolve. Thus, using entity information, in theory, could assist event coreference; and (b) the reverse situation, whereby the entity mentions are difficult for coreference resolution, but their respective, associated events are easy to resolve. Using event information could assist entity coreference. Therefore, we wish to represent our sentences in a manner that allows for both entity and event information to be organically captured and represented, without needing to explicitly define manually-created features like past systems (e.g., a boolean value if event mentions have governing dependency relations with the same lexical head).

5.2 Adding Sequential Context

Before we attempt to explicitly capture and use entity information, we first address the related, encompassing issue of using context better. Our CCNN compares two mentions' context in *exact* order – regardless of the convolutional kernel size, any given instance of the kernel is comparing the exact sub-embedding space of one mention to another. For example, if we were using a context-window size of three, then our system starts by using a filter over the word that appears three words before mention_1, and three words before mention_2.

The New Orleans Saints *relegated* Reggie Bush to the injured list on Wednesday.

Saints *put* Bush on I.R.

One of the key suspected Mafia bosses arrested yesterday *has incriminated* himself.

Police said Lo Presti *had incriminated* himself.

Figure 5.1: Example sentences from the ECB+ corpus, illustrating the theoretic benefits of jointly resolving both entities and events. Entity mentions are denoted by colored boxes, and ones with the same colors are coreferent. Event mentions are denoted by an italicized font. The top two sentences serve as an example of when entity coreference is easy but event coreference is challenging — the words *put* and *relegated* can have drastically different meanings. The bottom two sentences serve as an example of when entity coreference is difficult but event coreference is easy.

Although CNNs are particularly good at being robust to pattern variations, even sentences that convey identical meaning may often have drastically different lexical representations; sometimes the important words may appear three words before a mention in one sentence, but perhaps two words in another sentence. Moreover, the exact words themselves may differ between two sentences. Thus, we argue that the lexical distortion in sentences with coreferent event mentions is simply too large.

For example, let c_1 represent the context for mention m_1, and let c_2 represent the context for m_2. For any two words $w_1 \in c_1$ and $w_2 \in c_2$, where $w_1 = w_2$ (lexically), we simply measure the absolute difference in context position. Specifically, in Figure 5.2, the co-referring event mentions are highlighted blue, and the context positions are listed with red numbers. The useful word *Lohan* changed from being position -1 to position -8 (a difference of 7), and *People* changed from 8 to -11 (a difference of 19).

Among all event mention-pairs in the training set, only 11% of pairs are coreferent. When concerning only the coreferent pairs, we can construct context windows of five words (on both sides of the mentions). Most (74%) of coreferent mentions' context words do not appear in both mentions' contexts; in other words, only 26% of the mention pairs' collective context words appear in *both* mentions' contexts. Of this 26%, we can compute the distribution of the word position differences (i.e., lexical distortion), which we illustrate in Figure 5.3. Naturally, if the distances were all zero, our CCNN would have no trouble leveraging the relevant contextual words. If distances are greater than zero, or if the words are even much different between contexts, our CCNN will have less of a signal to use.

In an attempt to remedy this linear sensitivity to word order, we run a Bidirectional LSTM model on our corpus, with the hope that it is more robust to wide-range context and less sensitive to word reordering. Specifically, after running the LSTM, we extract the concatenated 300-length hidden layer embeddings for

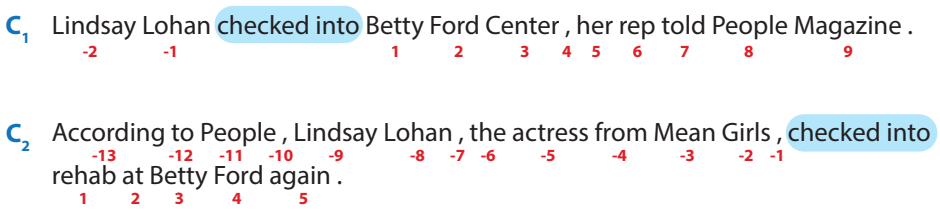


Figure 5.2: Example of common lexical distortion (word re-ordering) that exists within sentences.

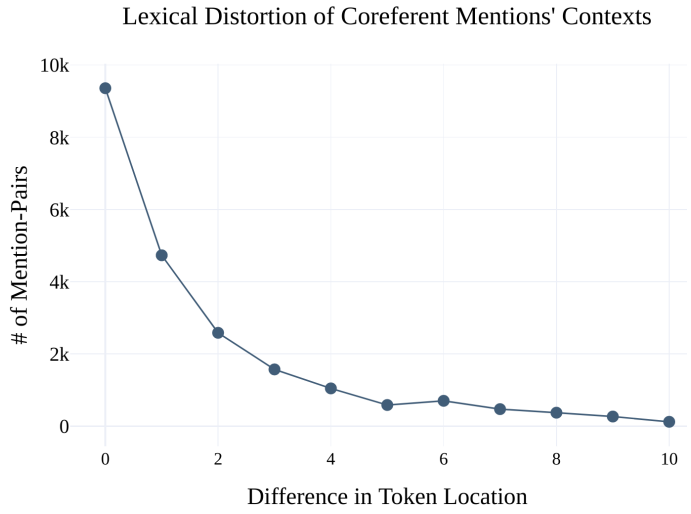


Figure 5.3: Distribution of lexical distortion (word re-ordering) distances between relevant words in coreferent sentences. Among coreferent event mention pairs, 74% of their context words do not exist within *both* mentions' context. This graph corresponds to the 26% of words that do.

	Within-Document				Cross-Document			
	MUC	B^3	CEAF	CoNLL F1	MUC	B^3	CEAF	CoNLL F1
SameLemma$_{any}$	40.4	66.4	66.2	57.7	66.7	51.4	46.2	54.8
HDDCRP [108]	53.4	75.4	71.7	66.8	73.1	53.5	49.5	58.7
Choubey [20]	62.6	72.4	71.8	68.9	73.4	61.0	56.5	63.6
FFNN+AGG	61.6	73.6	69.1	68.1 (0.14)	74.8	55.3	60.2	63.4 (0.21)
FFNN+NC	62.5	73.2	70.8	68.8 (0.17)	76.1	56.0	60.4	64.2 (0.18)
CCNN+AGG	65.2	74.2	69.0	69.5 (0.16)	75.8	55.8	62.7	64.8 (0.21)
CCNN+NC	67.3	73.3	69.6	70.1 (0.20)	77.2	56.3	62.0	65.2 (0.22)
CCNN+NC (ensemble)	**67.7**	**73.6**	**69.8**	**70.4 (0.13)**	**78.1**	**56.6**	**62.1**	**65.6 (0.17)**
Bi-LSTM-CCNN+NC	**63.4**	**72.0**	**68.9**	**68.1 (0.2)**	**75.3**	**57.2**	**58.1**	**63.5 (0.21)**

Table 5.1: Coreference Systems' clustering performance on the ECB+ test set, using the predicted mentions and testing procedure from Choubey and Huang [20]. Same results as reported in Table 4.6, but appended with results of having used Bi-directional LSTM embeddings + character embeddings as input to our CCNN

each token in our corpus. We use these embeddings, along with the character embeddings, as input to our CCNN (thus replacing GloVe's pre-computed lemma embeddings). Unfortunately, this does not offer any improvement, as reported in Table 5.1. We also experimented with supplementing our best-performing lemma + character embedding features with our newly-created LSTM embeddings; however, performance decreased slightly.

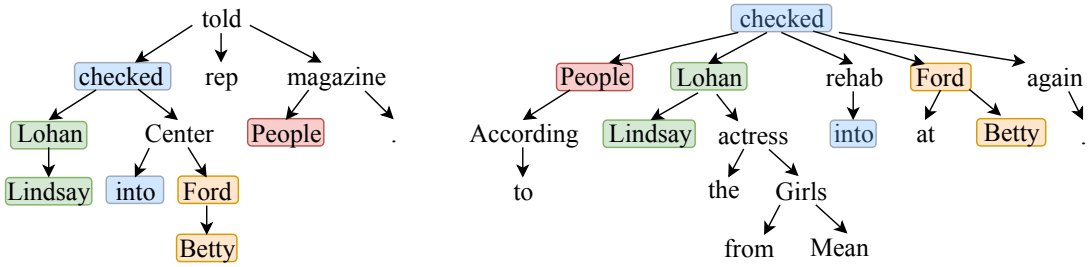
5.3 Structured Context

5.3.1 Motivation

In our daily lives, human language is represented sequentially (e.g., in verbal communication and in written text, such as our corpus); however, the underlying structure is compositional, so it is often useful to represent language as hierarchical, tree-like structures for computer models. While the reasons and methods for constructing such structures are well-studied by linguists [57], at the very least, NLP may benefit from tree structures due to the: (1) hierarchical representations allowing for increasing levels of abstraction [8, 93]; and (2) allowance of exploiting compositional units [95, 96, 98].

5.3.2 Dependency Parse Trees

As mentioned in Section 2.3.3, dependency parse trees model each sentence with respect to which words depend on which, and the connections are directed, unordered, and labelled by a relation. Given two sentences that contain co-referent events, their linear representations may be wildly different (as discussed), but their



Lindsay Lohan checked into Betty Ford Center, rep told People Magazine.

According to People, Lindsay Lohan, the actress from Mean Girls, checked into rehab at Betty Ford again.

Figure 5.4: Dependency parse trees for two sentences that contain co-referent event mentions *checked into* and *checked into*, illustrating the similar structure, despite their sequential sentence representations being significantly different. Mentions that are coreferent with each other are displayed with the same colored boxes.

dependency parse trees will likely have less variability. For example, in Figure 5.4 we see two sentences from the previous example have similar structures, including: the coreferent events *checked into* and *checked into* both have dependent children *Lindsay Lohan* and *Betty Ford*.

5.3.3 TreeLSTMs

In an attempt to capture meaningful embeddings for each word (node) based on its placement in the dependency parse tree, we turn to TreeLSTMs [98]. In short, TreeLSTMs are similar to traditional LSTMs in that they aim to capture representations of each word unit based on the context, while being able to handle long-range dependencies. However, instead of being restricted to linearly ordered data, it is designed to work with tree structures. Specifically, we use the *Child-Sum TreeLSTM* variant, which works as follows: We represent each sentence by its dependency parse tree, which was parsed by Stanford Core NLP [70]. Each word j is represented as a node in the tree. Akin to traditional LSTMs, each node j has a corresponding input vector x_j, along with:

- an input gate i_j
- an output gate o_j
- a hidden state h_j
- a memory cell m_j

Unlike traditional LSTMs, our TreeLSTM's gating and memory cell updates are dependent on every child node. Related, it has one forget gate f_{jk} for each child node k, which allows for selectively using information

from each child. Our input vectors x_j are initialized with the same GloVe embeddings that we used in our CCNN experiments. For any particular node j, let $C(j)$ denote its set of children. The TreeLSTM's transition functions are defined as:

$$\begin{aligned}
\tilde{h}_j &= \sum_{k \in C(j)} h_k \\
i_j &= \sigma(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}) \\
f_{jk} &= \sigma(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}) \\
o_j &= \sigma(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}) \\
u_j &= \tanh(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}) \\
c_j &= i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k \\
h_j &= o_j \odot tanh(c_j)
\end{aligned} \tag{5.1}$$

Fortunately, dependency parse trees are naturally rooted with event mentions — verbs are the top-most governing word token, on which all other words depend. Thus, for any two event mentions in our corpus, we can construct two TreeLSTMs, one for each event's corresponding sentence. The roots of the trees can then be measured in terms of their similarity, and we can assert that more similar roots correspond to coreferent event mentions. To enforce this property, we set our target to be a multi-class prediction $\hat{y}$ corresponding to if the two event roots are coreferent or not[1], defined by:

$$\begin{aligned}
h_\times &= h_L \odot h_R \\
h_+ &= |h_L - h_R| \\
h_s &= \sigma(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}) \\
\hat{p}_\theta &= \text{softmax}(W^{(p)} h_s + b^{(p)}) \\
\hat{y} &= r^T \hat{p}_\theta
\end{aligned} \tag{5.2}$$

In words, $\hat{y}$ is a similarity score based on the distance and angle between the two root nodes' hidden states/embeddings. This can ultimately be viewed as a conjoined model, as there is only one unique TreeLSTM model, and both copies share tied weights and work toward the single task of coreference prediction. The objective function measures the KL-divergence between the predicted $\hat{y}$ multi-class score and the gold truth's

[1] denoted by [0,1] or [1,0] for being coreferent or not, respectively.

score y:

$$J(\theta) = \frac{1}{m}\sum_{k=1}^{m} \text{KL}(p^{(k)} \| \hat{p}_\theta^{(k)}) + \frac{\lambda}{2}\|\theta\|_2^2 \tag{5.3}$$

where m represents the total number of training pairs, and k denotes the k^{th} pair of sentences.

5.4 Event Coreference

5.4.1 Experiments

We wish to use our Conjoined Child-Sum TreeLSTM to predict if two event mentions are coreferent or not. Since our model is defined to work with predicting only root nodes, this would leave us without predictions for all other event mentions that are at non-root locations (many sentences contain more than one event). However, when the network updates its weights, this includes updating all children's weights, not just the root nodes. For this reason, at testing time we can compare any two event mentions, regardless of their location in the tree, by using their *hidden states*. Specifically, our prediction score is the cosine similarity between the hidden states h_a and h_b:

$$\text{cosine similarity}(h_a, h_b) = \frac{\sum_{i=1}^{n} h_{a_i} h_{b_i}}{\sqrt{\sum_{i=1}^{n} h_{a_i}^2}\sqrt{\sum_{i=1}^{n} h_{b_i}^2}} \tag{5.4}$$

When an event mention has multiple tokens, we follow suit with our previous experiments by summing all embeddings in place. At training time, we are limited to using only the pairs of *sentences* that have dependency parse tree roots that are annotated by our corpus as being event mentions; although all roots should be event mentions, the ECB+ corpus does not comprehensively, perfectly annotate every word token, and since we need gold truth data to know if the roots should co-refer or not, this limits us to using a subset of the training data. Specifically, in Table 5.2 we list the amount of ECB+ data available for use in our various experiments.

5.4.2 Results

While we hoped that a tree structure representation would yield improved performance and not need *any* explicit features, the pairwise mention results in Table 5.3 demonstrate this is not the case. This model used no character embeddings or any explicit signal/features beyond the pre-trained GloVe word embeddings.

	Within-Document			**Cross-Document**		
	# Mention Pairs	# Mention Pairs with Entity Paths	# Unique Sentences	# Mention Pairs	# Mention Pairs with Entity Paths	# Unique Sentences
Train	5,922	2,960	580	73,925	38,625	730
Dev	775	428	117	8,939	4,980	137
Test	8,512	3,947	794	87,415	42,466	958

Table 5.2: ECB+ corpus statistics. When representing each sentence as a dependency parse tree, not every event mention has an entity as a child. The number of event mention pairs that do are listed as "# Mention Pairs with Entity Paths." Our Conjoined TreeLSTM model operates on pairs of sentences, and the training set for it only considers sentences that are rooted at event mentions that are annotated in our corpus.

	Within-Document		Cross-Document	
	Development (39 pairs)	Test (285 pairs)	Development (1,004 pairs)	Test (8,190 pairs)
SameLemma$_{any}$	52.63	69.93	54.51	57.52
CCNN$_{avg}$	55.22	70.88	56.87	59.01
CCNN$_{ens}$	54.55	71.34	56.93	58.96
TreeLSTM	76.92	58.74	57.02	53.36

Table 5.3: Models' mention-pair F1 performance after having all trained on the exact same subset of ECB+ Training: the event mentions that are roots of dependency parsed sentences. The number of *sentence* pairs on which we evaluated is listed in parenthesis for each corresponding data split.

From Table 5.2, we can see that our training size is relatively small, especially considering that we only trained on unique sentence pairs. Figure 5.5 shows the effects of the training size, whereby it is clear that our model would improve with more data. Related, our model is optimizing to predict pairs of sentence trees based on their roots, so when the weights are adjusted via backpropagation, nodes deep within the tree (e.g., near leaf nodes) will naturally receive a weaker signal from the objective function. Figure 5.6 illustrates this weakness, as event mentions that exist at lower depths tend to have much lower accuracy (root mentions have a depth of one, and each successive child has an increased depth number). Note, the diagonal has strong performance because those sentence pairs are nearly identical with each other, and some the pairs at the lowest depth have very few mention pairs (e.g., only 1 pair), so those results are less significant. Nonetheless, a trend exists in that lower depth yields worse performance.

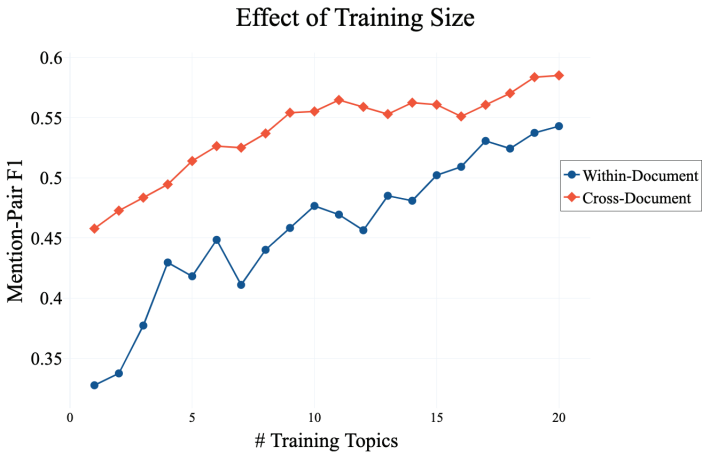


Figure 5.5: TreeLSTM's Mention-Pair F1 performance on the ECB+ Test Set while varying the training size.

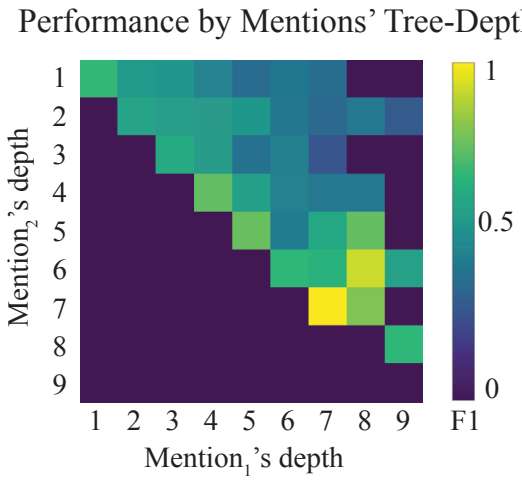


Figure 5.6: Mention-Pair F1 performance on the ECB+ Test Set, measured on a per-depth basis. A Depth of 1 corresponds to event mentions at the root of trees. The lower-depth pairs are sparse, with only a few pairs for representation.

	Within-Document	Cross-Document
P(event_coref)	0.11	0.17
P(entity_coref)	0.30	0.33
P(event_coref | entity_coref)	0.24	0.34
P(entity_coref | event_coref)	0.65	0.68

Table 5.4: Given dependency parse tree representations of the ECB+ corpus, we compute coreference statistics for event mentions and their associated, dependent entities. Using gold annotations, we see a symbiotic relationship whereby knowledge of entities coreference strongly increases the likelihood of event coreference, and vice versa.

5.5 Event + Entity Coreference

5.5.1 Motivation

Based on the previous experiments' results, along with findings from the previous chapter, we are motivated to: (1) create/use more training data; (2) improve coreference of mentions at lower tree-depth; and (3) explicitly incorporate entity information.

In an attempt to estimate the potential benefits from using entity information toward the task of event coreference, we can compute basic statistics and probabilities from the ECB+ Training Set. In particular, let us continue with using the dependency parse trees for every sentence. For every given event mention in the tree, we can traverse down through all of its dependents until we reach an entity mention. If there are multiple entity mention dependents, we pick the one with the lowest tree depth (we define the event mention to have a depth of zero). If there are multiple entity mentions at the same, minimal depth, we use all of that exist at this minimal depth. We consider this entity/entities to be associated with the event mention. For example, in Figure 5.7, the top tree has an event *checked into*, and its associated entity is *Lindsay Lohan* because it has a depth of one, whereas the other descendent entities in the tree have a higher depth (e.g., *Betty Ford* has a depth of two, and the entity *People* is not reachable and thus not considered). The bottom tree has an event *checked into*, and it has three associated entities because all three of them have the same, minimal depth of one: *People*, *Linsday Lohan*, and *Betty Ford*.

Now, given any two event mentions, we can compare their respective entity mentions, with the hope that there will be a correlation between the likelihood of entities being coreferent and their corresponding events being coreferent. This limits us to using only the ECB+ sentences that contain annotations for entities and events. Using the gold truth ECB+ annotations, Table 5.4 illustrates that there exists a strong correlation. Specifically, for the within-document mentions, the apriori probability that any two events are coreferent is

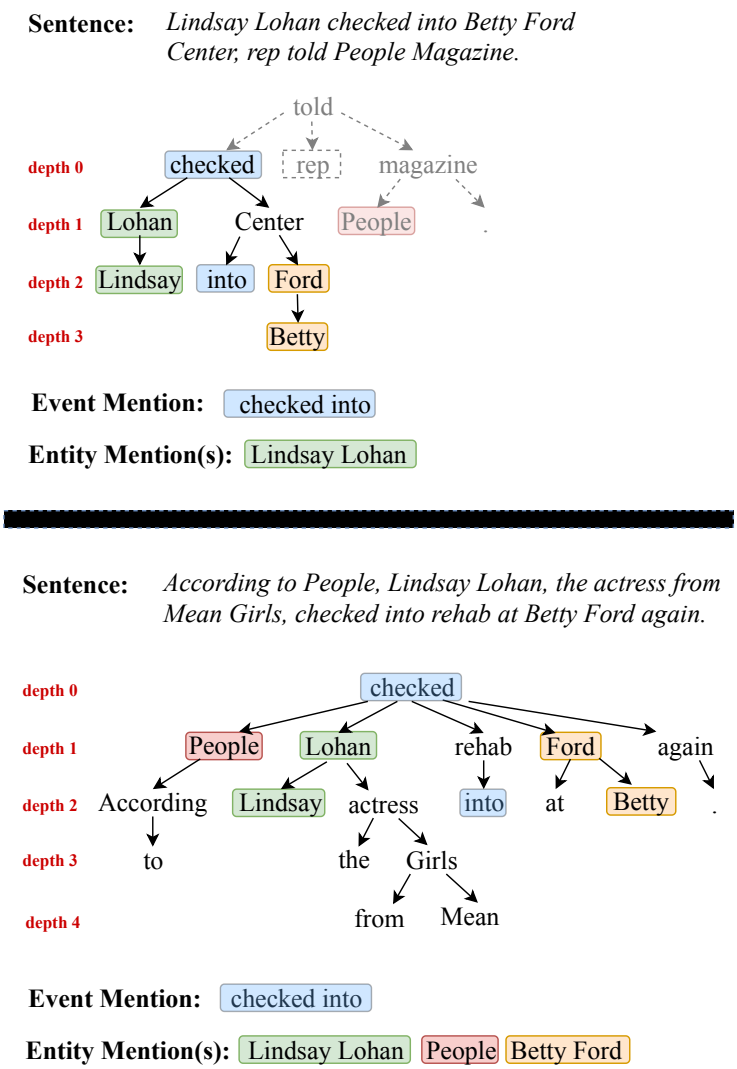


Figure 5.7: Dependency parses for two sentences, illustrating that for any given event mention, its associated entity mention(s) are only those that are reachable via the shortest path (i.e., the minimal depth). In the bottom example, three entities tie for having a depth of one, so all are used.

0.11. If we consider the entities that are associated with events, there is a 0.30 probability that the entities are coreferent. However, if we look at only the pairs for which entities co-refer, their governing events are coreferent with a probability of 0.24. Thus, the knowledge of entities being coreferent increases the likelihood of their governing events being coreferent by more than twice as much. Similarly, given that two event mentions are coreferent, their dependent entities are more than twice as likely to be coreferent than if we had no knowledge of the events. This correlation exists for both within-document and cross-document environments.

5.5.2 Experiments

We address the first two motivating points by adjusting our training data. Instead of training only on the dependency parses of complete sentence pairs, we additionally train on every sub-tree that is rooted at an event mention. This provides more training data and ameliorates the issue of having poor performance on events that are located further down the tree (high depth numbers). Now, every event mention will be the root of its own TreeLSTM, both at training and testing time.

We now address the third motivating point: when performing event coreference, we create a function that is based on our CCNN's event-based prediction and the events' associated entity mentions' TreeLSTM embeddings:

$$f(event_1, event_2) = \alpha * s_1 + (1 - \alpha) * s_2 \tag{5.5}$$

Where s_1 is our CCNN's prediction for $event_1$ and $event_2$, and s_2 is the L^2 distance between the TreeLSTM's hidden embeddings corresponding to $event_1$'s associated entities and $event_2$'s associated entities. If any event has multiple entities (due to being at the same, minimal depth), we sum all of their embeddings in-place before computing the L^2 distance. Since we explicitly use events' associated entities, we are limited to using only those event mentions that have dependent entities, per the ECB+ annotations. Thus, these experiments are not directly comparable to past results, since it concerns a subset of the original data. Using the development set, we fine-tune all parameters, including α.

Last, in addition to using our CCNN for event coreference, then supplementing it with entity embeddings from the TreeLSTM, we performed coreference from the opposite perspective: we used our CCNN for entity coreference (which serves as a baseline), then we supplemented it with events embeddings from the TreeLSTM, which were then joined into the following equation:

	Within-Document		Cross-Document	
	Development (347 pairs)	Test (4,565 pairs)	Development (3,959 pairs)	Test (44,949 pairs)
SameLemma$_{any}$	50.00	48.39	60.16	47.03
CCNN$_{avg}$	62.58	51.91	67.95	54.05
CCNN$_{ens}$	63.41	51.90	69.23	54.39
CCNN$_{ens}$ + TreeLSTM	64.50	53.22	70.17	54.87

Table 5.5: Event coreference mention-pair F1 performance for various models, all of which were trained on the exact same subset of ECB+ Training: event mentions that have dependency paths to entities. The number of event-mention pairs on which we evaluated is listed in parenthesis for each corresponding data split.

	Within-Document	Cross-Document
	Test (13,679 pairs)	Test (48,291 pairs)
SameLemma$_{any}$	46.84	43.95
CCNN$_{avg}$	48.11	44.20
CCNN$_{ens}$	48.87	44.36
CCNN$_{ens}$ + TreeLSTM	50.40	45.93

Table 5.6: Entity coreference mention-pair F1 performance for various models, all of which were trained on the exact same subset of ECB+ Training: entity mentions that are dependent on annotated events. The number of entity-mention pairs on which we evaluated is listed in parenthesis for each data split.

$$f(entity_1, entity_2) = \alpha * s_1 + (1 - \alpha) * s_2 \tag{5.6}$$

Where s_1 is our CCNN's prediction for $entity_1$ and $entity_2$, and s_2 is the L^2 distance between the TreeLSTM's hidden embeddings corresponding to $entity_1$'s governing event and $entity_2$'s governing event.

5.5.3 Results

In Table 5.5, we demonstrate that using entity information improves event coreference, both on a within-document and cross-document basis. Similarly, in Table 5.6, we show that using event information improves entity coreference. Note, the number of entity-mention testing pairs differs from the number of event-mention testing pairs simply because there are more entity nodes than event nodes. We remind the reader that an event may often have multiple entities as dependent children, yet we test every pair of event mentions exactly once, independent of its number of entities.

5.6 Discussion

In this chapter, we illustrated the need for incorporating context better, the theoretical potential of using structured representations, and the empirical benefits of leveraging such, along with using information from both entities and events. Entity coreference was not the focus of our research, so we believe there is further room for improvement toward this task – especially regarding pronoun resolution.

For both entity and event coreference, the mention-pair results only show a small improvement for the cross-document setting. We attribute this to the nature of the corpus, whereby many sentences in differing documents are nearly identical with one another. Thus, just using lemma embeddings of the event mentions yields high performance. Further evidence of this is made clear when one looks at the true positives and false negatives of our original CCNN event coreference system: of the false negative event pairs, there is no longer a strong correlation between entity and event coreference like we saw in Table 5.4. The event pairs that our CCNN correctly predicted (true positives) maintained the aforementioned correlation; yet, the ones we missed did not. So, leveraging entity information had limited potential in *further* improving our system.

Naturally, in other real-life scenarios, documents are free to discuss anything and non-coreferent event mentions may often use the same verbs. For example, if two randomly selected documents used the word *announced*, there should not be a strong apriori probability that the two event mentions, *announced*, are coreferent with each other. If a corpus with more varied content were created, we strongly believe that the benefits of using entity information would be even further pronounced.

Furthering this line of work, there are a few areas we believe could offer promise: in addition to jointly using entity and event information, there might be benefits of jointly *resolving* both entity and event coreference. One can imagine an iterative process that converges in its estimates of coreference, while membership is being successively refined. Perhaps using latent variable models would be a sound choice. Very recent advances in language modelling, namely via Transformers [27], have illustrated outstanding ability in using natural language context. Since coreference hinges on intelligently factoring in nuanced, long-term context dependencies, we believe Transformers would be fruitful to use. Just a few days ago, new research has shown strong progress in representing tree structures with RNNs [95]. While coreference resolution is an incredibly difficult and large problem, we believe our work, along with the new models discussed here, holds the strongest promise for continued advances: that is, eliminating hand-crafted features, representing language via tree structures, wisely handling context, and using both entity and event information – while ideally jointly resolving both.

Chapter 6

Conclusions

In this dissertation, we researched event coreference resolution for both within-document and cross-document settings. We surveyed literature and outlined weaknesses in existing systems, which we then systematically addressed. Unlike current models, which commonly rely on dozens of hand-engineered features, we developed a CCNN mention-pair model that yields state-of-the-art results while using only two features: lemma embeddings and character embeddings. We provided an analysis of other useful features (word embeddings, dependency parents/children, and part-of-speech tags). Further, we identified weaknesses in the de facto standard agglomerative-based clustering approaches, which we improve upon with our more holistic, neural-based clustering (NC). Namely, instead of merging clusters on a per-mention basis, we learn to merge on a cluster-basis, which is more robust spurious mentions and prevents clusters from growing erroneously large.

Having thoroughly examined our results and categorized our errors, we were motivated to further improve our system by using context better. In particular, we noted that structured representations can offer significant benefits over traditional, linear representations of text. We used dependency parse trees to represent our sentences, and we used Conjoined TreeLSTMs to model the likelihood of any two event mentions being coreferent. Having noted that performance decreases as we concern event mentions that are at lower depths in the dependency trees, we enhanced our TreeLSTM approach by modelling sub-trees. This allows for a more direct feedback signal from our objective function, as every event mention becomes the root of its own sub-tree. Last, we used the hidden tree embeddings of entities, in conjunction with our CCNN model, to further improve our coreference performance. Thus, we demonstrate a symbiotic relationship by combining entities and events to achieve better results than that from any individual model.

Appendix A

Appendix

The following is a sample of three documents from Topic 1 of the ECB+ corpus. We have removed the sentences that were not annotated. Words that comprise a mention are surrounded by [ ] brackets, and prefaced with a unique identify (e.g., ent8) signifying if they are an entity or event. The number part of the prefaced identifier has no meaning other than denoting which mentions co-refer with each other.

Document: 1_1ecbplus.xml

Title: ent8[Lindsay Lohan] v195[leaves] ent10[Betty Ford] , v196[checks into] ent197[Malibu rehab] .

First published : June 13 , 2013 4 : 59 pm EDT .

Body: ent8[Lndsay Lohan] has v195[left] ent10[the Betty Ford Center] and is v196[moving] to ent197[a rehab facility in Malibu , Calif] . , ent198[Access Hollywood] has v199[confirmed] . A ent200[spokesperson] for ent201[the Los Angeles Superior Court] confirmed to ent198[access] that a ent202[judge] v203[signed] an order ent204[yesterday] v205[allowing] the v196[transfer] to ent197[Cliffside] , ent197[where] ent8[she] will v206[continue] with her ent18[90 - day] court - mandated v19[rehab] . ent8[Lohan] ' s ent37[attorney , Shawn Holley] , v207[spoke] out about the v196[move] . " ent8[Lindsay] is v208[grateful] for the treatment ent8[she] v181[received] ent10[at the Betty Ford Center] . ent8[She] has v209[completed] ent8[her] course of treatment there and v210[looks forward] to v206[continuing] ent8[her] treatment and v211[building] on the foundation v212[established] ent10[at Betty Ford] , " ent37[Holley] v213[said] in a v214[statement] to ent198[Access] . The ent8[actress] v9[checked into] ent10[the Betty Ford Center] in ent215[May] as part of a v216[plea] v217[deal] v218[stemming from] ent8[her] ent86[June 2012] ent219[car] v87[accident] v25[case] .

Document: 1_2ecbplus.xml

Title: May 2 , 2013 , 1 : 12 pm . ent56[Lawyer] : ent8[Lindsay Lohan] v11[checks into] ent12[rehab facility] . Updated 4 : 08 p . m . ET .

Body: ent8[Lindsay Lohan's] ent56[attorney] v61[said] ent132[Thursday] the ent8[actress] v11[checked into] a ent12[southern California rehab facility] ent12[that] a state official said is v70[unlicensed] to v240[perform] the type of v19[treatment] a ent241[judge] v242[required] ent8[her] to v181[receive] . ent56[Mark Jay Heller] v243[told] a judge that ent8[Lohan] was v244[settling in] at ent12[Morningside Recovery , a treatment facility in Newport Beach] . "My ent8[client] is v62[ensconced] in the ent63[bosom] of that ent12[facility] right now , " ent56[Heller] v228[argued] after a ent55[prosecutor] v229[objected] to ent8[Lohan's] v145[choice] of ent136[rehab facilities] . ent8["She's] in ent12[rehab] right now . Nothing bad is v245[going to happen] . " . ent123[TMZ] v122[reported] ent8[Lohan] was v73[shopping] at an ent246[electronics store] while ent8[her] ent56[attorney] was in ent247[court] , and that ent8[she] never v248[entered] ent12[Morningside] . ent177[White] said he was "completely v249[blindsided"] by ent8[Lohan's] v11[placement] at ent12[Morningside] because ent56[Heller] had previously v235[agreed] to v250[send] the ent8[actress] to a different ent237[facility] that had been v238[vetted] .

Document: 1_1ecb.xml

Body: Another day in Hollywood ; another star in rehab . Word comes from ent5[People Magazine] and other celebrity news outlets that ent0[Tara Reid] , 33 , ent0[who] v193[starred] in " American Pie " and appeared on U.S. TV show " Scrubs , " has v1[entered] ent2[the Promises Treatment Center in Malibu , California] - the same ent2[facility] that in the past has been the rehab facility of choice for many a hollywood star . ent5[People] v194[said] Reid 's representative ent3[Jack Ketsoyan] v4[confirmed] the ent0[actress] 's v1[stay] ent2[at Promises] .

Bibliography

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING*, 2018. Cited on [4]

[2] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, Yiming Yang, James Allan Umass, Brian Archibald Cmu, Doug Beeferman Cmu, Adam Berger Cmu, Ralf Brown Cmu, Ira Carp Dragon, George Doddington Darpa, Alex Hauptmann Cmu, John Lafferty Cmu, Victor Lavrenko Umass, Xin Liu Cmu, Steve Lowe Dragon, Paul Van Mulbregt Dragon, Ron Papka Umass, Thomas Pierce Cmu, Jay Ponte Umass, and Mike Scudder Umass. Topic detection and tracking pilot study final report. In *In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, 1998. Cited on [1]

[3] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In *ACL*, 2015. Cited on [16]

[4] Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Granada, 1998. Cited on [40]

[5] Amit Bagga and Breck Baldwin. Cross-document event coreference: Annotations, experiments, and observations. In *Proceedings of the Workshop on Coreference and Its Applications*, CorefApp '99, pages 1–8, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics. Cited on [17]

[6] Cosmin Adrian Bejan and Sanda Harabagiu. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1412–1422, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. Cited on [18, 19, 20]

[7] Yoshua Bengio. Neural net language models. *Scholarpedia*, 3(1):3881, 2008. Cited on [14]

[8] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. Cited on [46]

[9] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66, 1994. Cited on [15]

[10] Eric Bengtson and Dan Roth. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 294–303, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. Cited on [23]

[11] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *In Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997. Cited on [4]

[12] Christopher M. Bishop and Nasser M. Nasrabadi. Pattern recognition and machine learning. *J. Electronic Imaging*, 16:049901, 2007. Cited on [7]

[13] Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. *CoRR*, abs/1603.06021, 2016. Cited on [16]

[14] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 737–744. Morgan-Kaufmann, 1994. Cited on [27]

[15] Eugene Charniak. *Introduction to deep learning*. MIT Press, 2018. Cited on [7, 9, 11, 13, 16]

[16] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*, 2018. Cited on [11]

[17] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jian Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*, 2015. Cited on [24]

[18] Jason P. C. Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370, 2016. Cited on [4]

[19] Do Kook Choe and Eugene Charniak. Parsing as language modeling. In *EMNLP*, 2016. Cited on [11]

[20] Prafulla Kumar Choubey and Ruihong Huang. Event coreference resolution by iteratively unfolding inter-dependencies among events. In *EMNLP*, 2017. Cited on [18, 19, 22, 23, 25, 30, 33, 41, 42, 46]

[21] Kevin Clark and Christopher D. Manning. Deep reinforcement learning for mention-ranking coreference models. *CoRR*, abs/1609.08667, 2016. Cited on [17, 19]

[22] Kevin Clark and Christopher D. Manning. Improving coreference resolution by learning entity-level distributed representations, 2016. cite arxiv:1606.01323Comment: Accepted for publication at the Association for Computational Linguistics (ACL), 2016. Cited on [5, 17, 19]

[23] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, 2007. Cited on [4]

[24] Agata Cybulska and Piek Vossen. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA), 2014. Cited on [20, 22]

[25] Agata Cybulska and Piek T. J. M. Vossen. Translating granularity of event slots into features for event coreference resolution. In *EVENTS@HLP-NAACL*, 2015. Cited on [18, 19]

[26] Naomi Daniel, Dragomir Radev, and Timothy Allison. Sub-event based multi-document summarization. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*, HLT-NAACL-DUC '03, pages 9–16, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. Cited on [1]

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. Cited on [11, 56]

[28] Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan R. Salakhutdinov. Gated-attention readers for text comprehension. In *ACL*, 2017. Cited on [11]

[29] Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *ACL*, 2015. Cited on [24]

[30] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2010. Cited on [12]

[31] Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. In *EMNLP*, pages 1971–1982. ACL, 2013. Cited on [5]

[32] Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490, 2014. Cited on [17, 19]

[33] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In *EMNLP*, 2018. Cited on [11]

[34] B. Fasel. An introduction to bio-inspired artificial neural network architectures. *Acta neurologica Belgica*, 103 1:6–12, 2003. Cited on [8]

[35] Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. A language-independent neural network for event detection. In *ACL*, 2016. Cited on [4]

[36] Frederic Brenton Fitch. Mcculloch warren s. and pitts walter. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics , vol. 5 (1943), pp. 115133. *Journal of Symbolic Logic*, 9:49–50, 1944. Cited on [8]

[37] Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. A convolutional encoder model for neural machine translation. In *ACL (1)*, pages 123–135. Association for Computational Linguistics, 2017. Cited on [24]

[38] Jeremy Getman, Joe Ellis, Stephanie Strassel, Zhiyi Song, and Jennifer Tracey. Laying the groundwork for knowledge base population: Nine years of linguistic resources for tac kbp. In *LREC*, 2018. Cited on [20]

[39] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288, September 2002. Cited on [4]

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. Cited on [7, 11, 16]

[41] Ruslan Salakhutdinov Gregory Koch, Richard Zemel. Siamese neural networks for one-shot image recognition. In *ICML*, 2015. Cited on [27]

[42] Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. Cited on [4]

[43] Aria Haghighi and Dan Klein. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 385–393, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. Cited on [17, 19]

[44] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. Cited on [10]

[45] Tian He. Coreference resolution on entities and events for hospital discharge summaries. In *Thesis*, 2007. Cited on [18]

[46] Sepp Hochreiter. Investigations on dynamic neural networks. *Diploma, Technical University*, 91(1), 1991. Cited on [15]

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. Cited on [4, 15]

[48] Eduard H. Hovy, Teruko Mitamura, Felisa Verdejo, Jun Araki, and Andrew Philpot. Events are not simple: Identity, non-identity, and quasi-identity. In *ACL*, 2013. Cited on [2]

[49] Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. Event coreference for information extraction. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, ANARESOLUTION '97, pages 75–81, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. Cited on [1, 17, 18]

[50] Richard Johansson and Pierre Nugues. Lth: Semantic structure extraction using nonprojective dependency trees. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 227–230, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. Cited on [4]

[51] Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd Edition). `https://web.stanford.edu/~jurafsky/slp3/`, 2019. [Online; accessed 25-March-2019]. Cited on [7]

[52] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *ACL*, 2014. Cited on [13]

[53] Kian Kenyon-Dean, Jackie Chi Kit Cheung, and Doina Precup. Resolving event coreference with supervised representation learning and clustering-oriented regularization. In **SEM@NAACL-HLT*, 2018. Cited on [18, 19, 23, 24, 25]

[54] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751. ACL, 2014. Cited on [24, 28]

[55] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. Cited on [12, 30]

[56] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *ACL*, 2018. Cited on [11]

[57] Hilda Koopman, Dominique Sportiche, and Edward P. Stabler. An introduction to syntactic analysis and theory. In *An Introduction to Syntactic Analysis and Theory*, 2013. Cited on [46]

[58] Sebastian Krause, Feiyu Xu, Hans Uszkoreit, and Dirk Weissenborn. Event linking with sentential features from convolutional neural networks. In *CoNLL*, 2016. Cited on [37]

[59] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. In *None*, 2005. Cited on [12]

[60] Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 489–500, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. Cited on [18, 19]

[61] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics, 2017. Cited on [3, 17, 19]

[62] Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*, 2018. Cited on [17, 19]

[63] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL*, 2014. Cited on [16]

[64] Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *ACL*, 2014. Cited on [4]

[65] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 507–516, New York, New York, USA, 20–22 Jun 2016. PMLR. Cited on [30]

[66] Jing Lu and Vincent Ng. Joint learning for event coreference resolution. In *ACL*, 2017. Cited on [18, 19]

[67] Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. Joint inference for event coreference resolution. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 3264–3275, 2016. Cited on [18, 19]

[68] Xiaoqiang Luo. On coreference resolution performance metrics. In *HLT/EMNLP*, 2005. Cited on [40]

[69] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics. Cited on [4]

[70] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. Cited on [27, 47]

[71] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL*, 2014. Cited on [18]

[72] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. Cited on [4]

[73] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. Cited on [14]

[74] Nafise Sadat Moosavi and Michael Strube. Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *ACL*, 2016. Cited on [39]

[75] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007. Publisher: John Benjamins Publishing Company. Cited on [4]

[76] Srini Narayanan and Sanda Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. Cited on [1]

[77] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. Joint event extraction via recurrent neural networks. In *HLT-NAACL*, 2016. Cited on [4]

[78] Thien Huu Nguyen and Ralph Grishman. Event detection and domain adaptation with convolutional neural networks. In *ACL*, 2015. Cited on [24]

[79] Thien Huu Nguyen, Adam Meyers, and Ralph Grishman. New york university 2016 system for kbp event nugget: A deep learning approach. In *TAC*, 2016. Cited on [24]

[80] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996. Cited on [8]

[81] Haoruo Peng, Yangqiu Song, and Dan Roth. Event detection and co-reference with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 392–402, 2016. Cited on [4, 18, 19]

[82] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. Cited on [15, 27]

[83] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, 2018. Cited on [11, 17]

[84] Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 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), pages 30–35, 2014. Cited on [40]

[85] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Comput. Linguist.*, 34(2):257–287, June 2008. Cited on [4]

[86] W.V. O. Quine. Events and reification. In *Action and Events: Perspectives on the philosophy of Donald Davidson*, pages 162–171, 1985. Cited on [2]

[87] Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Daniel Jurafsky, and Christopher D. Manning. A multi-pass sieve for coreference resolution. In *EMNLP*, 2010. Cited on [17, 19]

[88] Altaf Rahman and Vincent Ng. Coreference resolution with world knowledge. In *ACL*, 2011. Cited on [18]

[89] L. F. Rau. Extracting company names from text. In *Proc. of the Seventh Conference on Artificial Intelligence Applications CAIA-91 (Volume II: Visuals)*, pages 189–194, Miami Beach, FL, 1991. Cited on [4]

[90] Anna Rumshisky and Yuanliang Meng. Context-aware neural model for temporal information extraction. In *ACL*, 2018. Cited on [4]

[91] Gerard Salton, Aloysius Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975. Cited on [14]

[92] Sunita Sarawagi and William W Cohen. Semi-markov conditional random fields for information extraction. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1185–1192. MIT Press, 2005. Cited on [30]

[93] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, 61:85–117, 2015. Cited on [12, 46]

[94] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2017. Cited on [11]

[95] Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. *Proceedings of ICLR*, 2019. Cited on [46, 56]

[96] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013. Cited on [46]

[97] Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Comput. Linguist.*, 27(4):521–544, December 2001. Cited on [5]

[98] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015. Cited on [46, 47]

[99] Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 589–596, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. Cited on [4]

[100] Shyam Upadhyay, Nitish Gupta, Christos Christodoulopoulos, and Dan Roth. Revisiting the evaluation for cross document event coreference. In *COLING*, 2016. Cited on [3]

[101] Shyam Upadhyay, Nitish Gupta, Christos Christodoulopoulos, and Dan Roth. Revisiting the evaluation for cross document event coreference. In *COLING*, 2016. Cited on [39]

[102] Marc B. Vilain, John D. Burger, John S. Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *MUC*, 1995. Cited on [39]

[103] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015. Cited on [11]

[104] Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. Learning global features for coreference resolution. In *HLT-NAACL*, pages 994–1004. The Association for Computational Linguistics, 2016. Cited on [5, 17, 19, 26]

[105] Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. Learning anaphoricity and antecedent ranking features for coreference resolution. In *ACL (1)*, pages 1416–1426. The Association for Computer Linguistics, 2015. Cited on [5, 17, 19]

[106] Travis Wolfe, Mark Dredze, and Benjamin Van Durme. Predicate argument alignment using a global coherence model. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings*, 2015. Cited on [2]

[107] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan R. Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. Cited on [11]

[108] Bishan Yang, Claire Cardie, and Peter I. Frazier. A hierarchical distance-dependent bayesian model for event coreference resolution. *TACL*, 3:517–528, 2015. Cited on [18, 19, 22, 23, 25, 30, 37, 41, 46]

[109] Xiang Yu and Ngoc Thang Vu. Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages. *CoRR*, abs/1705.10814, 2017. Cited on [24]

[110] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jian Zhao. Relation classification via convolutional deep neural network. In *COLING*, 2014. Cited on [24]

[111] Rui Zhang, Cícero Nogueira dos Santos, Michihiro Yasunaga, Bing Xiang, and Dragomir R. Radev. Neural coreference resolution with deep biaffine attention by joint mention detection and mention clustering. In *ACL*, 2018. Cited on [3]

[112] Rui Zhang, Cicero Nogueira dos Santos, Michihiro Yasunaga, Bing Xiang, and Dragomir Radev. Neural coreference resolution with deep biaffine attention by joint mention detection and mention clustering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 102–107. Association for Computational Linguistics, 2018. Cited on [17, 19]

[113] Yingjie Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *IJCNLP*, 2017. Cited on [13]