Towards the Extraction of Cross-sentence Relations through Event Extraction and Entity Coreference

Dissertation
zur Erlangung des akademischen Grades eines
Doktors der Philosophie
der Philosophischen Fakultät
der Universität des Saarlandes

vorgelegt von

Iliana Simova

aus Blagoevgrad, Bulgarien

Saarbrücken, 2021

Dekan: Prof. Dr. Augustin Speyer
Berichterstatter: Prof. Dr. Alexander Koller
Prof. Dr. Dietrich Klakow
Tag der letzten Prüfungsleistung: 29 November, 2021

Abstract

Cross-sentence relation extraction deals with the extraction of relations beyond the sentence boundary. This thesis focuses on two of the NLP tasks which are of importance to the successful extraction of cross-sentence relation mentions: event extraction and coreference resolution.

The first part of the thesis focuses on addressing data sparsity issues in event extraction. We propose a self-training approach for obtaining additional labeled examples for the task. The process starts off with a Bi-LSTM event tagger trained on a small labeled data set which is used to discover new event instances in a large collection of unstructured text. The high confidence model predictions are selected to construct a data set of automatically-labeled training examples. We present several ways in which the resulting data set can be used for re-training the event tagger in conjunction with the initial labeled data. The best configuration achieves statistically significant improvement over the baseline on the ACE 2005 test set (macro-F1), as well as in a 10-fold cross validation (micro- and macro-F1) evaluation. Our error analysis reveals that the augmentation approach is especially beneficial for the classification of the most under-represented event types in the original data set.

The second part of the thesis focuses on the problem of coreference resolution. While a certain level of precision can be reached by modeling surface information about entity mentions, their successful resolution often depends on semantic or world knowledge. This thesis investigates an unsupervised source of such knowledge, namely distributed word representations. We present several ways in which word embeddings can be utilized to extract features for a supervised coreference resolver. Our evaluation results and error analysis show that each of these features helps improve over the baseline coreference system's performance, with a statistically significant improvement (CoNLL F1) achieved when the proposed features are used jointly. Moreover, all features lead to a reduction in the amount of precision errors in resolving references between common nouns, demonstrating that they successfully incorporate semantic information into the process.

Ausführliche Zusammenfassung

Die ständig wachsenden Mengen an digitalen Daten, die in unstrukturierter Form vorliegen, haben das Interesse an Technologien geweckt, die in der Lage sind, automatisch nützliche, in ihnen kodierte Informationen zu destillieren. Informationsextraktion ist ein Bereich der natürlichen Sprachverarbeitung, der sich mit der Entdeckung und Strukturierung solcher in natürlichsprachigem Text gefundenen Informationen beschäftigt.

Die Informationen, die für die Informationsextraktion von Interesse sind, haben typischerweise die Form von Fakten über Entitäten und Beziehungen zwischen ihnen. Beispiele für solche Beziehungen sind die Verwandtschaftsbeziehungen (z. B. Eltern, Geschwister, Ehepartner) zwischen zwei Entitäten des Typs "Person", die in biografischen Texten häufig vorkommen, oder die Beziehung zwischen zwei Firmenentitäten, die besagt, dass die eine die andere übernommen hat. Das Erkennen von semantischen Beziehungen zwischen Entitäten aus unstrukturiertem Text fällt in den Bereich der Aufgabe Relationsextraktion.

Das Thema Relationsextraktion ist die Hauptmotivation und der Ausgangspunkt für die in dieser Arbeit vorgestellten Experimente. Genauer gesagt, geht es um die Extraktion von Relationen mit unterschiedlichen Komplexitäten (definiert durch die Anzahl ihrer Relationsargumente), die über mehrere Sätze hinweg ausgedrückt werden, oder "cross-sentence" Relationsextraktion (Abschnitt 1.2). In dieser Arbeit wird der Standpunkt eingenommen, dass das Problem der satzübergreifenden Relationsextraktion aus mehreren Bestandteilen besteht. Es muss eine Analyse auf Dokumentenebene durchgeführt werden, die die beiden Hauptmechanismen der Verbreitung beziehungsrelevanter Informationen adressiert, nähmlich wiederholte Verweise auf Ereignisse und auf Entitäten.

Unsere Untersuchung der satzübergreifenden Relationserwähnungen im cockrACE-Korpus [Krause et al., 2014] ergab, dass 90,5% der Erwähnungen mindestens einen anaphorischen Verweis auf Relationsargumente enthalten, die an einem anderen Ort im Dokument realisiert werden. In solchen Fällen kann die Aufgabe der Koreferenzauflösung (coreference resolution), die sich auf die Zuordnung von Erwähnungen gleicher Entitäten in Äquivalenzklassen konzentriert, die Extraktion der fehlenden Informationen ermöglichen.

Für den Zugriff auf beziehungsrelevante Informationen, die über ein Dokument verstreut sind, schlagen [Krause et al., 2016], die Aufgabe der Ereignisverknüpfung zu nutzen. Ereignisse sind textuelle Beschreibungen von Ereignissen in der realen Welt und deren Teilnehmern. Die Aufgabe der Ereignisextraktion besteht darin, solche Erwähnungen zu erkennen und ihre Teilnehmer in Bezug auf die Rolle zu klassifizieren, die sie im Ereignis spielen. Die Ereigniskoreferenz (oder Ereignisverknüpfung) ist eine nachfolgende Aufgabe, die sich auf die Gruppierung von Erwähnungen desselben Ereignisses konzentriert. Durch die Verknüpfung von Erwähnungen desselben Ereignisses erhalten wir oft Zugang zu zusätzlichen Ereignisteilnehmern für dieses Ereignis.

Die Ereigniskoreferenz allein ist nicht ausreichend, um auf alle Relationsargumente zuzugreifen, da anaphorische Referenzen vorhanden sind. Daher ist die Auflösung der Koreferenz ein integraler Bestandteil des Prozesses. Außerdem wird bei der Durchführung von Ereignisverknüpfungen im Allgemeinen davon ausgegangen, dass Ereigniserwähnungen als Voraussetzung erkannt werden. Die Ereignisextraktion an sich ist jedoch keine triviale Aufgabe. Die Datenarmut stellt ein Problem dar, welches den Einsatz von semi-überwachtem Lernen erfordert. In dieser Arbeit konzentrieren wir uns genauer auf dieses Problem. Wir schlagen vor, die Datenbeschränkungen durch den semi-überwachten Algorithmus "self-training" [Abney, 2007] zu lösen. Im Gegensatz zu anderen häufig verwendeten semi-überwachten Ansätzen ist "self-training" nicht inhärent argumentzentriert und kann daher Erwähnungen von Ereignissen unabhängig vom Vorhandensein von Ereignisargumenten im selben Satz erkennen.

Der erste Teil der Arbeit befasst sich mit der Problematik der Datenarmut bei der Ereignisextraktion. Neuronale Ansätze, die auf die Verfügbarkeit einer ausreichenden Menge an annotierten Ereignisdaten angewiesen sind, stellen den aktuellsten Stand der Technik bezüglich Ereignisextraktion dar (Abschnitt 3.3.2). Bestehende Korpora für diese Aufgabe sind in Bezug auf Größe und Abdeckung der Ereignistypen begrenzt. Es hat sich gezeigt, dass das am weitesten verbreitete Korpus, das ACE 2005 [Walker et al., 2006], eine Reihe von Problemen bezüglich der Datenarmut aufweist (Abschnitt 3.2). Das Fehlen einer ausreichenden Menge an Trainingsdaten und die Kosten, die mit dem Aufbau von menschlich annotierten Korpora verbunden sind, haben mehrere Arbeiten dazu inspiriert, Methoden zur automatischen Extraktion zusätzlicher vorklassifizierten Trainingsbeispiele zu erforschen (Abschnitt 3.4)

Der erste Beitrag dieser Arbeit ist ein Ansatz für die automatische Extraktion von zusätzlichen vorklassifizierten Daten für die Ereignisextraktion durch selftraining. Wir schlagen vor, die Aufgabe der Identifikation und Klassifikation von Ereignisauslösern als ein Sequenz-Labeling-Problem zu betrachten und definieren einen Baseline Bi-LSTM Ereignis-Tagger. Der Selbsttrainings-Prozess beginnt mit dem Training dieses Ereignis-Taggers auf dem ACE 2005-Korpus und verwendet ihn, um aus einer großen Sammlung von unstrukturiertem Text neue gelabelte Beispiele für die Aufgabe zu extrahieren. Die Modellvorhersagen mit hohem Vertrauen werden ausgewählt, um einen Datensatz mit automatisch beschrifteten Trainingsbeispielen zu erstellen.

Um die Fehler des Taggers so gering wie möglich zu halten, verwenden wir eine Filterstrategie und behalten nur die Klassifizierungsentscheidungen bei, die einen bestimmten Schwellenwert des Vertrauens überschreiten. Hohe Schwellenwerte führen dazu, dass nur die vertrauenswürdigsten Modellentscheidungen beibehalten werden. Diese entsprechen in der Regel den am häufigsten auftretenden Ereignistypen in den ursprünglichen Trainingsdaten, führen aber oft dazu, dass weniger häufige Ereignisbeispiele vollständig herausgefiltert werden. Um dies abzumildern, verwenden wir einen dynamischen Schwellenwert: wir beginnen mit den vertrauenswürdigsten Entscheidungen des Modells und gehen dann zu einem niedrigeren Vertrauensschwellenwert zurück, bis wir die gewünschte Anzahl von Beispielen für jeden Ereignistyp erreicht haben.

Wir stellen mehrere Möglichkeiten vor, wie der resultierende Datensatz in Verbindung mit den ursprünglich vorklassifizierten Daten zum erneuten Training des Event-Taggers verwendet werden kann. Beim ersten Augmentierungsansatz, aug_{concat}, erhalten die neu extrahierten Beispiele die gleiche Wichtigkeit wie der ursprüngliche Trainingsdatensatz, während die übrigen drei Strategien darauf abzielen, den Effekt der Augmentierungsdaten beim erneuten Training zu verringern. Bei der gleichmäßigen Skalierungsparameter-Augmentation ($\text{aug}_{uniform}$) wird beim Training auf dem Augmentationsdatensatz eine Skalierungskonstante von 0,1 verwendet. Bei aug_{weight} wird der Verlust, der beim Training mit einem Stapel aus überwiegend identischen Ereignissen entsteht, durch die Häufigkeit dieser Ereignisse im Trainingssatz geteilt durch eine Konstante skaliert. In diesem Setup werden häufig auftretende Ereignistypen stärker von ihren entsprechenden Augmentationsbeispielen beeinflusst, mit der Annahme, dass sie aufgrund ihrer höheren Häufigkeit zu qualitativ besseren Augmentationsdaten beitragen. Bei $\text{aug}_{rev.weight}$

wird der gegenteilige Effekt angestrebt: häufige Ereignisse erhalten wenig Hilfe von den Augmentationsdaten, da sie bereits im ursprünglichen Trainingssatz gut vertreten sind, während seltene Ereignistypen beim Training einen höheren Augmentationsschub erhalten.

Die Evaluation unserer Daten-Augmentierungsansätze auf dem Testset des ACE-Korpus zeigt, dass die beste Augmentierungsstrategie "$\text{aug}_{uniform}$" ist. Diese Konfiguration erreicht eine statistisch signifikante Verbesserung gegenüber der Baseline auf dem ACE 2005 Testset (Makro-F1), sowie in einer 10-fachen Kreuzvalidierung-Evaluation (Mikro- und Makro-F1). Die Fehleranalyse zeigt, dass der Augmentierungsansatz vor allem bei der Klassifikation der im ursprünglichen Datensatz am stärksten unterrepräsentierten Ereignistypen von Vorteil ist.

Der zweite Teil der Arbeit beschäftigt sich mit dem Problem der Koreferenzauflösung. Die Auflösung von Referenzen ist eine herausfordernde Aufgabe, da ein referenzierender Ausdruck oft mehrere konkurrierende Antezedenten-Kandidaten hat, die ähnliche Oberflächeneigenschaften aufweisen, wie z.B. Übereinstimmung in Zahl und Geschlecht. Während ein gewisses Maß an Präzision durch die Modellierung von Oberflächeninformationen über Entitätserwähnungen erreicht werden kann, hängt deren erfolgreiche Auflösung oft von semantischem oder Weltwissen ab.

Eine lange Reihe von Arbeiten widmet sich der Kodierung semantischer und enzyklopädischer Informationen in Form von Features für die überwachte Koreferenzauflösung (Abschnitt 5.5.2). Einige der untersuchten Quellen für solche Informationen sind die lexikalische Datenbank WordNet [Daumé III and Marcu, 2005; Ponzetto and Strube, 2006; Bengtson and Roth, 2008], Wikipedia und verwandte enzyklopädische Datenbanken [Strube and Ponzetto, 2006; Ratinov and Roth, 2012; Rahman and Ng, 2011] und das Mining unstrukturierter Daten [Haghighi and Klein, 2009; Bansal and Klein, 2012]. Selektive Präferenzen und semantisches Parsing wurden zur Pronomenauflösung eingesetzt [Ponzetto and Strube, 2006; Rahman and Ng, 2011; Heinzerling et al., 2017]. Die in diesem Teil der Diplomarbeit vorgestellten Experimente untersuchen eine alternative Quelle von semantischem Wissen für die Auflösung von Koreferenzen, nämlich Worteinbettungen.

Worteinbettungen sind kontinuierliche, niedrigdimensionale Vektordarstellungen von Wörtern, die aus großen, unbeschrifteten Korpora gelernt wurden (Abschnitt 2.3). ähnlich wie bei Ansätzen der distributiven Semantik, die sich auf die Idee

stützen, dass der Sinn eines Wortes durch die Gesellschaft definiert werden kann, in der es steht, wird die Worteinbettung eines Tokens aus seinen Kontextwörtern gelernt. Dadurch werden Wörtern, die in ähnlichen Kontexten auftreten, ähnliche Repräsentationen zugewiesen. Die intrinsische Evaluation hat gezeigt, dass Worteinbettungen syntaktische (z. B. Geschlecht) und semantische (z. B. Verwandtschaft und Ähnlichkeit) Eigenschaften von Wörtern erfassen (Abschnitt 5.6).

Diese Arbeit bietet einige Einblicke, wie Worteinbettungen auf die Aufgabe der Koreferenzauflösung angewendet werden können. Wir stellen drei verschiedene Merkmale vor, die aus Worteinbettungen abgeleitet werden. Dazu gehören (1) ein Embedding-Cluster-Merkmal, das einer semantischen Klasse gleicht, (2) eine Umgebung, bei der jede Dimension des Embeddings ein separates numerisches Merkmal ist, und (3) eine Reihe von Kosinus-Ähnlichkeitsmerkmalen, die kontextuelle Informationen einbeziehen. Das Einbettungs-Cluster-Merkmal kann für die Auflösung von Koreferenzen von Vorteil sein, da es sicherstellen kann, dass zwei koreferierende Erwähnungen semantisch kompatibel sind. Zum Beispiel wäre eine Erwähnung, die ein Fahrzeug bezeichnet, nicht mit einem Referenten vom Typ "Person" kompatibel. Eine einfache Möglichkeit, Worteinbettungen für die Koreferenzauflösung zu verwenden, ist die direkte Einbeziehung des Einbettungsvektors. Jede Dimension kann als ein separat latentes Merkmal betrachtet werden, das verschiedene syntaktische oder semantische Eigenschaften eines Wortes kodiert [Turian et al., 2010]. Wir experimentieren auch mit kompakteren Repräsentationen des Worteinbettungsvektors, die durch die Hauptkomponentenanalyse gewonnen werden. Schließlich definieren wir Kosinus-ähnlichkeitsbasierte Merkmale, die darauf abzielen, mehr vom Kontext des verweisenden Ausdrucks in den Entscheidungsprozess einzubeziehen. Dies zielt auf Fälle ab, in denen der betreffende verweisende Ausdruck semantisch arm ist (z. B. ein Pronomen) und seine erfolgreiche Disambiguierung daher auf der Erforschung seines Satzkontextes beruht.

Unsere Evaluierungsergebnisse und die Fehleranalyse zeigen, dass jedes dieser Merkmale dazu beiträgt, die Leistung des Baseline-Koreferenzsystems zu verbessern. Die Verbesserung des CoNLL-F1-Scores durch den gemeinsamen Einsatz aller vorgeschlagenen Features war statistisch signifikant. Wir beobachteten eine Reduzierung der Gesamtzahl der Präzisionsfehler. Darüber hinaus führen alle Features zu einer Reduktion der Präzisionsfehler bei der Auflösung von Referenzen zwischen Substantiven. Diese Ergebnisse legen nahe, dass sie erfolgreich einige semantische Informationen in den Prozess einbeziehen.

Acknowledgements

Firstly I would like to thank Prof. Dr. Alexander Koller for his guidance and support during my PhD, and for instilling good scientific practices in me. I would like to thank Prof. Dr. Dietrich Klakow, with whom I've had the pleasure of working since my time as a Master student and who first sparked my interest in machine learning through his lectures. Thank you for agreeing to be my second supervisor. I would also like to thank Prof. Dr. Hans Uszkoreit, who provided me with the opportunity to pursue a PhD and who introduced me to the topic of relation extraction.

This thesis would not have been possible without the unwavering support of Sébastien and Yoana, who have been by my side through many tough moments in the past several years. Thank you for your patience, for believing in me and for pushing me to go forward.

Special thanks to the many colleagues and friends who have made my time at UdS special: Eran, Eli, Klára, Iona, Ingmar, Anna, Marc, Torsten, Claudia and Lucia. Thanks to Beata for the unforgettable moments and always being able to distract me from worrying too much about my PhD. Thanks to Julia for proof-reading German summary of the thesis which was a big help in the final stages of preparing this document.

Last but not least, I would like to express my gratitude to my family for always being there for me. A very special thanks to Kiril for encouraging me to pursue a PhD and supporting me along the way.

Contents

Chapter 1

Introduction

The continuously growing volumes of digital data available in unstructured form have prompted the interest in technologies capable of automatically distilling useful pieces of information encoded within them. Information Extraction (IE) is a field in Natural Language Processing (NLP) which focuses on the discovery and structuring of such information found in natural language text. What is meant by structuring is storing the extracted knowledge in a format which facilitates its further analysis and processing by people or other applications, not unlike entries in a data base. IE can assist higher-level NLP tasks dealing with natural language understanding, such as text summarization, question answering, dialogue systems, and more. It has been employed in the domains of business, law, medicine, finance and others [Gupta et al., 2020; Ghoulam et al., 2015; Nasar et al., 2018; Zhong et al., 2020]. Some examples sources of unstructured data for these domains include news reports, social media interactions, electronic patient records, and court rulings.

The information of interest in IE typically takes the form of facts about *entities* and *relations* between them. *Entities* can include mentions of persons, locations, dates, or other predefined types of concepts relevant for the task at hand. In the business domain, for instance, names of products might be of interest, while medical domain applications might focus on names of diseases and treatments. Examples of *relations* include the kinship (e.g., parent, sibling) or spousal relationships between two entities of type *person*, common in biographical text. Another example is the relation between two company entities, denoting that one has taken ownership of the other (the company acquisition relation). The extraction and

classification of such information falls in the scope of the task Relation Extraction (RE) (Section 1.1).

The topic of RE is a main motivation and starting point for the experiments presented in this thesis. More specifically, the extraction of relations of arbitrary complexities (defined in terms of number of their relation arguments), which are expressed across multiple sentences, or cross-sentence relation extraction (Section 1.2). In this thesis, we adopt the view that the problem of cross-sentence relation extraction is a composite one. Due to the complexity of this task, it is difficult to address it via a one-size-fits-all solution. Rather, an analysis on the document level has to be performed which addresses the two main mechanisms of spreading relation relevant information: repeated references to events and entities.

1.1 Relation Extraction

Relation Extraction aims at discovering semantic relations between entities from unstructured text. A relation can be represented as an ordered tuple of entities, the relation *arguments*. For instance, the relation between an entity of type person and an entity of type organisation which holds when a person is the founder of an organisation can be represented as *founder-of*<Person, Organisation>[1]. In practice, a relation is often defined in terms of a minimal set of at least two *obligatory* relation arguments, while additional relation-relevant information is treated as secondary or optional arguments which can be extracted if present in the relation mention [Krause, 2018]. For instance, minimally two entities of type person are necessary for expressing the spousal relation, but the starting and end date of a marriage are valuable[2] secondary arguments as they define the time period for which the relation holds.

A relation may be expressed in natural language in a variety of ways. Figure 1.1 presents different ways of expressing the spousal relation. A mention of the marriage *event* (1) expresses the relation, as it refers to the point in time from which the relation holds. Similarly, the renewal of vows indicates they were exchanged as part of a wedding ceremony in the first place (2). The divorce event (3) expresses

[1] A *relation mention* of a concrete instance of this relation, *founder-of*<Bill Gates, Microsoft>, can be found in the sentence "Bill Gates founded the software company Microsoft Corporation with his friend Paul Allen".

[2] For instance, for downstream tasks such as question answering.

(1) *Peter* and *Mary* **tied the knot** in *2000*. (*marriage* event)
(2) *Peter* and *Mary* **renewed their vows**. (*vow renewal* event)
(3) *Peter* and *Mary* **divorced** in *2005*. (*divorce* event)
(4) *Peter***'s wife** *Mary* attended the party.

FIGURE 1.1: Several different ways of expressing the spousal relation

the relation, as it *entails* that the two people were previously married. In (4), the relation is expressed via a noun phrase referring to the state of being married. Relations are often expressed through mentions of events: textual descriptions of real-world events and their participants[3].

The lack of labelled training data is a major challenge in RE. The target relations are usually dictated by the downstream application which RE serves: an application in the business domain would call for the extraction of very different relation types than those in the medical domain. Little to no training data may be available for a relation of interest. However, the success of a machine learning RE model is dependent on the availability of sufficient amounts of training examples which cover the variety of ways of expressing the relation in natural language. This has prompted research in approaches to RE which are more adaptable to new domains and relation types. To circumvent the expensive process of corpora annotation, these approaches typically employ a form of semi-supervised learning, such as *bootstrapping* [Brin, 1998; Agichtein and Gravano, 2000; Yangarber et al., 2000; Xu et al., 2007; Fujiwara and Sekine, 2011] or *distant supervision* [Mintz et al., 2009; Riedel et al., 2010; Krause et al., 2012b], which utilize small quantities of domain-specific knowledge to discover mentions of relations in large unlabeled corpora.

Bootstrapping is an iterative algorithm in which the knowledge acquired at every step serves as input to subsequent iterations (Figure 1.2). The process starts off with a small set of relation tuples referred to as *seeds*, which can be defined by hand based on prior knowledge, or found in structured knowledge bases, provided some exist. Next, occurrences of the seed are detected in a large collection of unstructured text. It is assumed that if the seed relation arguments co-occur

[3]Xu et al. [2007] define n-ary relations as events. Our investigation of the spousal relation mentions in the cockrACE corpus [Krause et al., 2014] confirmed that 94% of the relation mentions with more than two arguments are in fact events, as is the case with 61% of the binary relations. We will come back to a more in-debt discussion of events and the similarities and differences between event and relation extraction in Chapter 3.

in close proximity (usually within the same sentence) then the text is likely to expresses the given relation. The discovered candidate relation mentions are used to generalize relation extraction *rules* or patterns, which are in turn applied to extract previously unseen seeds. While the process relies on very little domain-specific knowledge (the initial seeds), it is prone to error propagation and semantic drift, and thus typically involves elaborate techniques for filtering the RE rules prior to rule application. The end-product of the bootstrapping loop includes both the extracted seeds and the learned RE rules. Bootstrapping RE rules are typically string patterns or paths in a syntactic tree [Sudo et al., 2001, 2003; Xu et al., 2007], such as the shortest path in a dependency parse of the sentence containing the relation mention which links all relation arguments (Figure 1.2). Path-based RE rules are meant to include the lexical items which express the relation of interest (here: "win", "prize") with place-holders for the relation arguments, ignoring small word order variations, and excluding what are considered less significant tokens outside the shortest path. However, such methods have the disadvantage of relying on possibly erroneous parses, and are limited to relations expressed within a single sentence.

Distant supervision is another semi-supervised approach which utilizes a knowledge base (e.g., Freebase [Bollacker et al., 2008]) as a source of supervision for learning a relation classifier from otherwise unlabelled data. Unlike bootstrapping, distant supervision is not iterative. The process typically starts off with a large set of seeds, and follows the same assumption that if seed arguments co-occur in the same sentence then the sentence is likely to express the seed relation. Any sentence which contains the seed arguments is used to extract features for that relation type, with a certain level of noise expected in the data [Mintz et al., 2009].

1.2 Cross-sentence Relation Extraction

When dealing with relations involving more than two arguments, the likelihood that some of the information is distributed in a broader context than a single sentence increases. A relation mention is *cross-* or *inter-sentential* if one would have to look across the boundaries of a single sentence in order to collect the complete information about the entities participating in it which is available in the text.

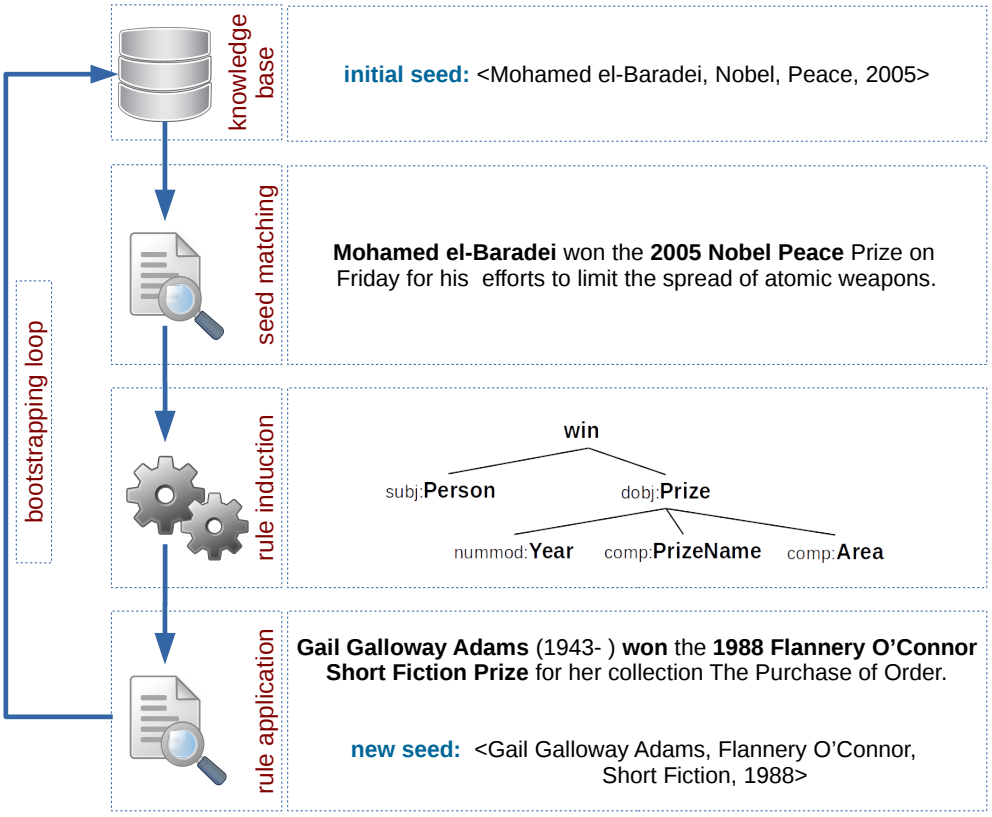

FIGURE 1.2: An example of a bootstrapping loop employed in [Xu et al., 2007] for a relation from the award winning domain with arguments <*person*, *prize name*, *prize area*, *year*>.

Cross-sentence relation extraction is an underexplored problem in IE. The most investigated domain (and associated relation types) for intra-sentential RE has been news-wire corpora. The news-wire domain is characterized by a redundancy of the reported facts, increasing the likelihood that a relation mention containing all of the relevant relation arguments can be found within a single sentence in at least one document. Going beyond the sentence boundary brings along complexity and the added benefit of doing it may be difficult to showcase in this situation. For certain more specialized domains which rely on less-redundant data, however, focusing on cross-sentence mentions can be of much importance. Data sources such as patient records, for instance, do not offer the luxury of repeated mentions of the same relation, but rather may contain a single occurrences of a relation instance, possibly spread across several sentences. In their work in the medical domain, Quirk and Poon [2017] report that performing cross-sentence extraction as opposed to single sentence extraction lead to a *doubled* yield of unique relations describing drug–gene interactions.

While existing labeled corpora for RE do contain inter-sentential relation mentions[4], there is the need for a large-scale RE corpus devoted to studying this problem[5]. The lack of such a data set poses another major challenge for developing solutions to the problem. Most of the existing works which perform semi-supervised learning report performance on their own distantly supervised data sets, which makes comparing them difficult.

Existing cross-sentence relation extraction approaches introduce extensions to traditional methods which enable the extraction of relations expressed over several *consecutive* sentences [Melli et al., 2007; Swampillai and Stevenson, 2011; Quirk and Poon, 2017; Peng et al., 2017]. A common trend is the construction of a graph representation of the text span and utilizing a path in this graph to guide the extraction process, inspired by the shortest path approach used in bootstrapping. Quirk and Poon [2017] present a distant supervision approach targeting the extraction of a binary relation expressing drug–gene interactions. If two entities of the correct types co-occur in a window of *up to three consecutive sentences* and are known to stand in the target relation according to a knowledge base, this text snippet is selected as a positive training example, while co-occurring entity pairs not present in the knowledge base and their contexts serve as negative examples. At the core of their system is a graph-based representation, referred to as a *document graph*. The edges of the graph incorporate the dependency parses of the adjacent sentences. Connections between them are established by linking their root nodes, adjacent tokens, as well as via discourse relations produced by a discourse parser, and entity coreference links between anaphora and their antecedents. Features consisting of paths in the graph connecting the two entities are extracted to train the relation extraction model. In a subsequent work, Peng et al. [2017] extend this approach to n-ary relations and utilize a graph long short-term memory (LSTM) network for modelling the connections in the same underlying document graph.

The main drawback of the proposed approaches for detecting cross-sentence relation mentions is that they are limited to a fixed context size of several consecutive sentences. In practice, relation arguments can be spread across arbitrarily large contexts[6]. The adoption of this limitation is necessary due to the nature of the

[4] Swampillai and Stevenson [2010] estimate that 9.4% of the binary relation mentions in the ACE 2003 corpus [Mitchell et al., 2004] are inter-sentential.

[5] This has been remedied post completion of the experiments in this thesis by the introduction of the DocRED corpus [Yao et al., 2019]

[6] For instance, via entity coreference: the distance between an anaphor and its antecedent can reach up to 30 sentences in extreme cases [Mitkov, 2002]

Then, two years ago, he met [Bonny Bakley]$_1$ at a club in Los Angeles. [..]

Born in Morristown, N.J., [Bakley]$_1$ came from a middle-class background.

By [her]$_1$ mid-20s [she]$_1$ was **married** to Paul Gawron, now 51, who worked as a laborer.

SPOUSE (she) SPOUSE (Paul Gawron)

FIGURE 1.3: Cross-sentence mention of spousal relation with two arguments of type person from the cockrACE corpus [Krause et al., 2014].

semi-supervised approach utilized for discovering relation instances. Distant supervision relies on the assumption that when entities of specific types co-occur in close proximity in an unstructured text, they likely participate in a given relation. The process is error-prone. To get a sense of the amount of error which can be expected in distantly-supervised data, Riedel et al. [2010] match Freebase relations against the New York Times (NYT) corpus [Sandhaus, 2008] and Wikipedia. They estimate that 38% of the sentences containing *nationality* seed matches, and 35% of the sentences containing *place-of-birth* seed matches against the NYT corpus do not in fact express the seed relation type, as is the case for 20% of the matches for both seed relation types against Wikipedia. While distant supervision is not inherently limited to relations expressed within a single sentence, increasing the allowed context size for matching the seed relation arguments is likely to bring substantial amounts of additional noise to the already noisy distantly supervised training data.

Consider the examples of cross-sentence mentions of the spousal relation in Figures 1.3 and 1.4. In Figure 1.3, the relation is expressed in the last sentence ("married"), but one of the relation arguments is only referenced locally. Extracting that "Paul Gawron" and "she" are spouses would not be valuable for downstream tasks. Moreover, a distant-supervision approach is likely to completely ignore this relation mention, as the process is guided by the presence of named entity relation arguments in the same sentence. Here a noun phrase coreference chain exists which links the referenced argument to its antecedent, the named entity "Bonny Bakley", carrying the full information on who the spouse is. Our investigation of the cross-sentential relation mentions in the cockrACE corpus revealed that 90.5% of the mentions contain at least one anaphoric reference to relation arguments realised in somewhere else in the document. The task of noun phrase Coreference Resolution

My wife and I were guests at a **wedding** on the Carnival Legend (LOCATION) on New Years Eve 2003 (DATE). [..] The **wedding** took place in the afternoon (TIME), before the ship set sail. Thus, the couple (SPOUSE) was actually **married** by a justice of the peace in the state of Florida (LOCATION).

FIGURE 1.4: Three coreferring mentions of a marriage event from the ACE 2005 corpus [Walker et al., 2006]. Each mention introduces additional relevant information via its event participants.

(CR), which focuses on assigning mentions of the same entities into equivalence classes, can enable the extraction of the missing information from arbitrarily large contexts in such cases.

Krause et al. [2016] propose *event coreference* as a way of extracting relation-relevant information on the document level. The task of *event extraction* (EE) deals with detecting such mentions and classifying their participants with respect to the role they play in the event. Event coreference (or *linking*) is a subsequent task which focuses on grouping mentions of the same event, similarly to how CR groups mention of the same entity. By linking mentions of the same event, we often gain access to additional event participants: Krause et al. [2016] found that approximately 23% of the event mentions in the ACE 2005 corpus lack arguments which are present in other mentions of the same event in the respective document. Consider the example in Figure 1.4. Each of the sentences discusses the same marriage ceremony. Successfully linking the three event mentions would tell us the location, date and time of the wedding, information which none of the mentions delivers when considered in isolation. Similarly to entity coreference, event coreference is performed on the document level, and thus no restrictions are placed on the distance between different mentions of the same event.

In this thesis, we adopt the view that the problem of cross-sentence relation extraction is a composite one. Due to the complexity of this task, it is difficult to address it via a one-size-fits-all solution. Rather, an analysis on the document level has to be performed which addresses the two main mechanisms of spreading relation relevant information: event and entity coreference. Event coreference on its own is not sufficient for accessing all relation arguments due to the presence of anaphoric references. Thus, coreference resolution is an integral part of the

process. Coreference resolution also benefits event coreference, as it can establish a connection to other mentions of the same event through repeated reference to their event participants.

The task of event linking assumes event mentions are detected as a prerequisite. However, event extraction in itself is not a trivial task. It suffers from similar data sparsity issues as relation extraction, and thus calls for the use of semi-supervised approaches. In this thesis we focus more closely on this problem. We propose to address the data limitations via a different semi-supervised algorithm: *self-training*. Unlike bootstrapping and distant-supervision, self-training is not inherently argument-centric and can thus be used to detect mentions of events regardless of the presence of event arguments in the same sentence.

1.3 Thesis Contributions

This thesis focuses on two of the NLP tasks which are central to the successful extraction of cross-sentence relation mentions, *event extraction* and *coreference resolution*.

Existing event corpora are limited in size and coverage of event types, which can be attributed to the domain specificity of the task. These limited resources in turn impact the performance of supervised event extraction models, and hinder their portability to new domains. The work described in the first part of the thesis demonstrates how the limited available resources can be used to enable the automatic extraction of additional training examples from unstructured text. We employ the technique self-training, in which a classifier trained on small amounts of labeled examples is used to extract additional training examples from unstructured data. We further propose several ways in which the automatically extracted examples can be utilized for augmenting the existing labeled data for the task.

Coreference resolution is another step crucial for enabling the extraction of information about relation arguments not explicitly stated in the sentence containing the relation triggers. A known challenge in coreference resolution is the need for lexical semantic or world knowledge to determine the semantic compatibility between different mentions of the same entity. Previous works have explored structured lexical semantic data bases and knowledge bases for the task, to mixed

success. The second thesis contribution focuses on this issue in coreference resolution. The presented experiments aim to answer whether word embeddings, which have been shown to encode certain semantic properties of words, can serve as a source of relevant information for the task and improve the precision of a CR model. We propose several ways of utilizing word embeddings in supervised coreference resolution and explore their usefulness in a detailed evaluation and error analysis.

1.4 Thesis Outline

The structure of the rest of the thesis is as follows:

Chapter 2 presents background in machine learning relevant for the experiments discussed in the remainder of the thesis. It begins with an introduction of semi-supervised learning and self-training. It then proceeds with a description of neural network architectures, focusing on models applicable to sequence labelling tasks. Both of these topics are of importance to the experiments described in Chapter 4. The chapter concludes with an overview of word embedding models, also referred to as distributed word representations. This topic is central to the experiments described in Chapter 6.

Chapter 3 focuses on the background and related works in event extraction. The chapter begins with a general introduction of events from a linguistic point of view, followed by an overview of event corpora. The next sections present the task of event extraction in greater detail, as well as state-of-the-art supervised event extraction models. The remainder of the chapter is dedicated to related works which employ semi-supervised approaches to the task.

Chapter 4 discusses our self-training approach to data augmentation for event extraction. The chapter begins with a more explicit definition of the problem we are focusing on, followed by a presentation of our baseline event tagger. The procedure for extracting additional training examples as well as the ways of utilizing these examples in conjunction with the existing training data are outlined next. The chapter concludes with a report on our evaluation results and error analysis.

Chapter 5 presents the background and related works in coreference resolution. It begins with a discussion of coreference as a linguistic phenomenon, outlining

the challenges associated with resolving co-referent mentions. The chapter proceeds with a presentation of existing coreference corpora. Supervised coreference resolution and the evaluation practices employed for the task are presented next, followed by a discussion of commonly employed hand-crafted features. The final sections state the importance of semantic knowledge for the task, and conclude with a discussion of why word embeddings are a suitable source of such knowledge.

Chapter 6 discusses the findings of our experiments which explore utilizing word embedding-based features for supervised coreference resolution. The chapter begins with an introduction of the ways in which we formulate features from word representations. A discussion of the experimental setup, including the coreference resolution model and word embedding models, follows. The chapter concludes with a presentation of the results and a detailed error analysis.

Chapter 7 concludes the thesis with a summary of the main contributions and future directions.

Chapter 2

Machine Learning Background

2.1 Self-training in Computational Linguistics

The following section describes the semi-supervised machine learning technique employed in the experiments described in Chapter 4. Section 2.1.1 provides a brief introduction to semi-supervised learning, followed by a description of the algorithm in Section 2.1.2, as well as some examples of its application in NLP.

2.1.1 Semi-supervised Learning

Machine learning algorithms can be broadly categorized as *supervised*, *unsupervised*, and *semi-supervised*, with respect to the type of data they utilise when learning to perform a specific task.

Supervised learning algorithms make use of labeled training examples, which consists of pairs of *inputs* and target *outputs* or *labels*. During training, these algorithms learn a mapping from inputs to outputs based on the properties of the data, with the ultimate goal of being able to predict the labels for previously unseen inputs at test time.

Supervised machine learning tasks include *classification* and *regression*. In the former, the output space consists of a finite set of labels or *classes*. For instance, assigning a "positive" or "negative" polarity label to a text snippet when performing sentiment analysis is a form of classification. Regression tasks, on the other hand, involve the prediction of continuous output values.

Let $X = \{x_1, .., x_n\}$ denote the input data consisting of n data points from a multi-dimensional space, also known as the feature vectors, and $Y = \{y_1, .., y_n\}$ denote their corresponding output labels. At training time, the model learns to predict the outputs associated with certain inputs in the training set $S_{train} = \{X_{train}, Y_{train}\}$, by minimising some task-specific error measure E_{train}. An example error measure is the *mean squared error* (2.1), suitable for a regression task. It computes the average squared difference between each predicted output value $\hat{y}$ and the actual output y.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{2.1}$$

The performance of the model on unseen examples is evaluated on a disjoint test set S_{test}, where X_{test} and Y_{test} are drawn from the same input–output pair distribution. The ability of the model to transfer performance from the training set to the test set is referred to as *generalization* [Graves, 2012], and is measured by computing the error measure E_{test}. Often an additional disjoint data set, S_{dev}, referred to as *validation* or *development* set, is used for the optimization of some of the model parameters during training, or to monitor the model for *overfitting*.

Overfitting occurs when the model has learned the training data so well, that is has memorised it rather than learned to generalize from it. Every variation or noise is construed as part of the underlying structure of the data and becomes part of the model. Thus, overfitted models perform perfectly at training time (E_{train} is very low), but fail with examples different than those present in the training set (E_{test} is high). On the opposite scale, *underfitting* can occur when the model is not able to adequately capture the underlying structure of the data due to insufficient parameters to model it, or insufficient training iterations to correctly adjust all model parameters. Such models can be seen as under-trained, as they do not manage to reach a sufficiently low E_{train} [Goodfellow et al., 2016].

The acquisition of labeled or gold-standard data for a learning problem requires expert knowledge. As this process is labor intensive and costly, such corpora are usually limited in size. Thus, the performance of supervised algorithms is often impacted by data sparsity issues.

In unsupervised learning tasks, the algorithms aim to build mathematical models from data which contains only inputs, but has no available target output labels.

Some examples of unsupervised learning include clustering or grouping of input data points according to some commonalities they exhibit. Clustering can be seen as the unsupervised equivalent of classification [Abney, 2007], however with several crucial differences. Firstly, the optimal number of groupings is often not known in advance, but is rather determined by the properties of the data itself, which is not the case for classification, where the number of desired classes are fixed. Secondly, even though clustering algorithms are able to create groupings of similar data points, they are not able to assign labels to them. Unsupervised learning is thus not directly applicable to solving the specific tasks of supervised learning guided by expert-made labeled data, but draws its strength from the ability to harness large quantities data available in an unannotated form. In addition to clustering, it has been utilised for discovering properties of data, feature engineering, and modelling probability densities of input data [Abney, 2007].

Semi-supervised algorithms are hybrid approaches aimed at overcoming the data sparsity issues faced by supervised methods with the help of unsupervised techniques. They learn from a mixture of labelled and unlabelled data. Typically task-specific labeled data is used to provide the characterization of the desired target output, while large quantities of unannotated data, which is cheaper to obtain, is used to provide additional information for improving the quality of the model.

Examples of semi-supervised learning include *self-training* and *co-training*, discussed in more detail in Section 2.1.2, and *distant supervision*, discussed in Section 1.1. All of the these approaches aim at automatically generating a labeled data set to be used later in a supervised learning setting, standalone or in addition to gold-standard data, if such is available.

2.1.2 Self-training

Self-training is a form of semi-supervised learning which leverages a model's own predictions to obtain new labeled examples for training. Abney [2007] provides the following general formulation of the algorithm:

Algorithm 1 Self-training

Given a set of labeled training examples L, and a set of unlabeled examples U

1: **procedure** SELFTRAIN(L, U)
2: $c \longleftarrow$ TRAIN(L)
3: **while** stopping criterion is not met **do**
4: $L \longleftarrow L +$ SELECT(LABEL(U, c))
5: $c \longleftarrow$ TRAIN(L)
6: **end while**
7: **end procedure**

The process relies on the availability of some labeled data set for the task of interest, L, which initially serves as training data for the model c. The resulting model is used to provide predictions on an unlabeled data set U. The most confident predictions are selected to become part of an extended labeled data set for re-training the model. This process can be repeated several times until a predefined stopping criterion is met.

The predictions of the model, $c(x)$, are usually in the form of a probability distribution over all possible output classes for each unlabeled example x in U, rather than hard labels. The probability of the most likely output label $\hat{y}$ can be seen as a measure of the confidence of the model's decision, and be used to determine whether the candidate is suitable for extending the labeled data set. In other words, if $P(\hat{y}|x) = \text{MAX}(c(x))$ is higher than a predetermined confidence threshold Θ, x can be selected for retraining with a pseudo-label $\hat{y} = \arg\max c(x)$ [Ruder and Plank, 2018].

Another possibility for the confidence measure is the ratio:

$$\frac{P(y_1|x)}{P(y_2|x)} \tag{2.2}$$

where y_1 is the most probable label, and y_2 is the second most probable label according to the model prediction [Abney, 2007]. This measure targets excluding cases where the classifier is hesitant between two predictions with close probability.

The data extraction process can be run for a predefined number of iterations, or stopped when the classifier and the extracted data set do not change anymore. Alternatively, a gold-standard validation set can be utilised to evaluate the performance of the classifier at each step and stop at an optimal configuration. However, running the algorithm in multiple iterations has the disadvantage that errors in the selection process can propagate and be amplified in subsequent iterations.

Abney [2007] presents two additional improvements to the selection process, namely *throttling* and *balancing*. Throttling refers to limiting the number of instances to be added to the labeled data at each iteration. Instead of accepting all instances whose confidence exceeds the threshold Θ, only the top k examples are selected, thus preventing a large influx of automatically labeled instances from overwhelming the influence of the existing labeled data. Balancing, on the other hand, makes sure that the number of new instances per class remains equal. This is important as the classifier's most confident decisions may be in favor of one class over the others, which, overtime can result in a more and more unbalanced data set as the iterations of the algorithm progress.

One of the earliest adaptations of the self-training in computational linguistics was by Yarowsky [1995], who focused on the task of word sense disambiguation. Self-training was used as a bootstrapping technique, starting off with a small amount of hand-picked training examples representative for the different senses of a polysemous word. The "one sense per discorse" property stating that the sense of word is highly consistent within any given document, was employed to augment and filter the list of candidates during selection, in addition to a confidence threshold.

Self-training has since been applied to a number of natural language processing tasks with varying degrees of success.

Several works have investigated the use of self-training for improving constituent parsing and for parser adaptation [McClosky et al., 2006b,a; Reichart and Rappoport, 2007].

McClosky et al. [2006a] utilise a two-phase parser-reranker system trained on the Wall Street Journal (WSJ) corpus to extract additional training examples from unannotated news articles. In their experiments they re-train the parser (but not the re-ranker) once, rather than in several iterations. They utilise a weighting scheme to give more importance to the original labeled data than the automatically extracted examples during retraining, as well as a throttling threshold when

selecting the amount of new examples. The best self-trained model achieves F-score of 92.1% on the WSJ corpus, with an absolute improvement of 1.1% over their baseline. In [McClosky et al., 2006b], the same approach is applied for parser adaptation, which is aimed at adapting a parser trained on data from a specific domain to another domain for which no labeled data is available. They achieve an error reduction of 28% on the Brown corpus test set using the same parser-reranker trained on WSJ.

Reichart and Rappoport [2007] also demonstrate the effectiveness of self-training for parser adaptation, but in a setup where less data is initially available for training, and without the use of a re-ranker. Besides the effect of the size of the initial training data (referred to as *seed*), they investigate the effect of the domains of the seed data and the unlabelled corpus used during self-training (referred to as self-training data). Rather than employing a confidence filter, the whole automatically annotated self-training corpus is combined with the seed for retraining, which is performed only once. When the seed data domain matches that of the test set (both WSJ), their model achieves large gains in both precision and recall over the baseline in all low-data conditions. If the self-training data is also in-domain, the gain in performance is slightly bigger. For the out-of-domain scenarios (WSJ as seed, Brown as test), the improvements in performance were in mainly in recall at the expense of precision, regardless of the domain of the self-training data.

Clark et al. [2003] report limited success of self-training for bootstrapping a Part Of Speech (POS) tagger. Another work in domain adaptation, [van der Goot et al., 2017], attempts to adapt a general purpose POS tagger to the domain of Twitter text, without performing any additional text normalization in the process. This is a challenging task, as such text contains a high number of misspellings and abbreviations which do not appear in standard corpora for POS tagging. The addition of new automatically-labeled examples via self-training did not show any consistent improvement.

Samad Zadeh Kaljahi [2010] employs self-training for the task of Semantic Role Labeling (SRL). He explores the effects of the parameters balancing and *pre-selection*. The former could be useful as the task is a multi-class classification with an unbalanced distribution of classes. The latter can be applied together with a modification of the self-training algorithm in which a subset (*pool*) of the unlabeled data is used at each iteration rather than the whole corpus [Abney, 2007]. Pre-selection aims to include those unlabeled instances into the pool which

can potentially be easier for the classifier to label correctly. Samad Zadeh Kaljahi [2010] selects examples based on a measure of simplicity of the sentence. The experiments reveal the usefulness of the balancing setting, which consistently improves over the baselines in different data conditions on both in-domain and out-of-domain evaluation sets. Pre-selection, on the other hand, only achieves marginal statistically insignificant improvements during evaluation.

Co-training [Blum and Mitchell, 1998] is another semi-supervised algorithm closely related to self-training. Its goal is similarly the expansion of an existing labeled data set via model predictions. Co-training, however, leverages two independent *views* of the labeled data in the process. For instance, in [Blum and Mitchell, 1998], where the task of interest is classification of web pages, the two views of the data are the words occurring on a page, and the words occurring in hyperlinks that point to that page. Two separate classifiers are trained on each view of the data, and then each model's predictions are used to enlarge the training set of the other. This strategy has the advantage that each classifier's predictions could potentially introduce valuable (and variable) new examples for training.

Algorithm 2 Co-training

Given a set of labeled training examples L, and a set of unlabeled examples U

procedure COTRAIN(L, U)
  while stopping criterion is not met **do**
    $c_1 \longleftarrow$ TRAIN(VIEW$_1$(L))
    $c_2 \longleftarrow$ TRAIN(VIEW$_2$(L))
    $L \longleftarrow L$ + SELECT(LABEL(U, c_1)) +
      SELECT(LABEL(U, c_2))
  end while
end procedure

Self-training can be seen as a special case of co-training with one classifier. Co-training can be further extended to the multiple classifier scenario. This allows for the use of voting strategies between the classifiers' predictions during the selection of candidates for re-training [Zhou and Li, 2005; Søgaard, 2010; Ruder and Plank, 2018].

The availability of different complementary descriptions of the data of interest is indispensable for the success of co-training. These can be induced by the use of several classifiers employing disjoint sets of features. For certain NLP tasks, however, such natural feature splits might be hard to determine. Ng and Cardie [2003] compare single-view (self-training) and multi-view (co-training) algorithms for the task of coreference resolution. In their experiments, self-training outperformed co-training under various parameter settings and was less sensitive to parameter changes.

Another approach which has been utilised to obtain different views of the data is training the same classifier several times on different sub-samples of the original labeled data [Zhou and Li, 2005; Ruder and Plank, 2018], if the initial seed is large enough to allow for sub-sampling. This strategy enables the voting between the classifiers' predictions, but it it unclear if this constitutes a real multi-view of the data as was intended in classical co-training. In fact, Ng and Cardie [2003] refer to this strategy as self-training with *bagging*.

2.2 Neural Sequence Labelling

This section is intended to provide some background in neural machine learning, necessary for describing the experiments in Chapter 4 and Chapter 6.

2.2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by the way in which biological brains process information. Their building blocks are the artificial neurons, or the nodes of the network, which are connected to each other via weighted edges. Information flows through the ANN similarly to how signals are transmitted via the synapses in the brain: each neuron receives and processes some input, and in turn passes its output to the neurons connected to it via outgoing edges. The weights of the edges, adjusted during the training of the model, represent the strength of the connections between the neurons. For instance, low weights correspond to weak synapses which would carry little of the signal over to the next neuron. In practice, the signal transmitted through the network is some real-valued representation of the input data.

An alternative way to view an ANN is as a complex mathematical function f, which maps a set of input values x to a set of output values y such that $y = f(x; W)$, where the values of the parameters W are chosen to result in the best function approximation [Goodfellow et al., 2016]. The function f is formed by composing many simpler functions, each of which provides its own representation of the input. Thus ANNs are able to build complex representations of the data in terms of simpler ones.

There exist a multitude of neural architectures which differ in terms of the arrangement of their neurons and the connections between them. One classification is with respect to cyclic connections: ANNs which do not allow for cycles are referred to as feed-forward neural networks (FFNN), while those which do are known as recurrent neural networks (RNN). The architecture employed in the experiments described in Chapter 4 is an example of the latter, and is presented in more detail in Section 2.2.4. We begin with a discussion of a simpler and widely used feed-forward architecture, the multi-layer perceptron (MLP), as a way to provide a formal introduction to neural networks as well as the background necessary for the discussion of the more complex architectures which follows.

2.2.2 Feed-forward Neural Networks

Figure 2.1 presents an example of a feed-forward network referred to as the multi-layer perceptron (MLP). The neurons of a MLP are arranged in layers, with connections transmitting the signal starting from the input layer, through one or more hidden layers, and finally to the output layer.

MLPs are said to be fully-connected, as each neuron in a given layer is connected to every neuron in the previous layer of the network. A neuron receives a linear combination of the outputs of the previous layer's nodes and the weights of their connecting edges W, including an optional bias parameter b. This value serves as input to a non-linear function ϕ, referred to as the *activation* or *squashing*[1] function. Such functions enable the network to learn complex, non-linear classification boundaries. ANNs are typically composed of multiple non-linear transformations of the input x, contributing to their great expressive power.

[1] they *squash* an infinite input domain into a finite output range [Graves, 2012]

Equation (2.3) summarizes the computation performed at each layer of a FFNN, with $l \in [1..L]$ denoting the index of the network layer, and $\cdot$ denoting the dot product operation.

$$h^{(l)} = \phi^{(l)}\Big(W^{(l)} \cdot h^{(l-1)} + b^{(l)}\Big) \tag{2.3}$$

In the running example (Figure 2.1), the input layer $h^{(0)} = x$ is the vector representation of the input data, $h^{(1)}$ denotes the single hidden layer of the network, $h^{(2)} = \hat{y}$ is the output layer, and the bias parameters are omitted for simplicity. Popular choices for the activation function at the hidden layer, $\phi^{(1)}$, include *tanh*, sigmoid, and relu.

The configuration of the network in terms of number of neurons and choice of activation function at the output layer is determined by the task in question. A convention for a classification task with $K>2$ classes is to have K output neurons and use a *softmax* (Equation (2.4)) function to obtain a probability distribution over the possible output labels [Graves, 2012]. On the other hand, binary classification can be carried out with a single neuron in the output layer and a *sigmoid* function, whose output can be interpreted as the probability of predicting one of the classes.

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{2.4}$$

In the following lines we will focus more closely on the data representation and computations performed when a MLP is applied to a multi-class classification scenario.

Let the training set S_{train} consist of input–output pairs of vectors (x, y). Each vector x constitutes some multi-dimensional real-valued representations of the input. In a multi-class classification problem, y takes the form of a one-hot vector representation of the output labels, where class k is encoded as a vector of size K containing "1" in position k, and "0" everywhere else. Such a representation not only serves the purpose of representing the categorical class variables, but is convenient during the optimization of the network, as it can be interpreted as a

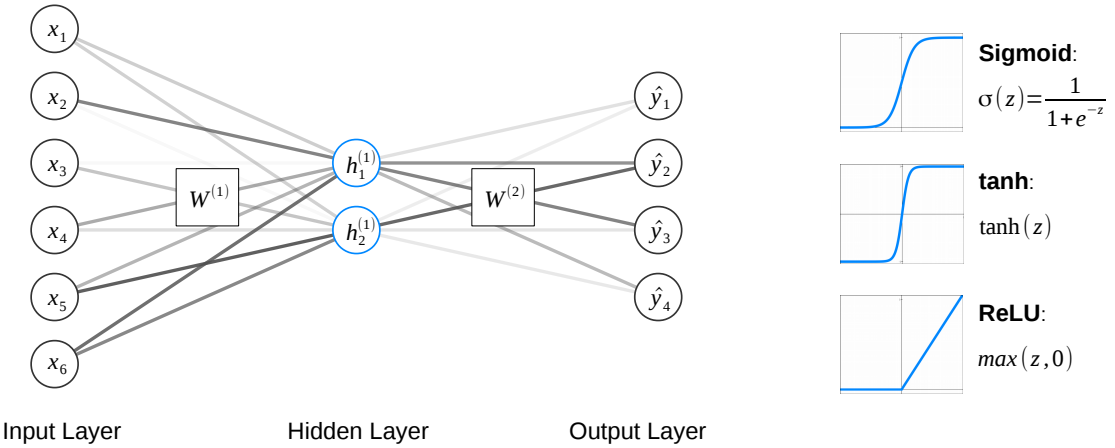

FIGURE 2.1: An example feed-forward neural network with one hidden layer. The color of the edges corresponds to the magnitudes of their associated weights. To the right are popular choices for the activation function of the hidden layer.

probability distribution over the possible output labels. Furthermore, we will use $\hat{y}$ to denote the output vector predicted by the network.

The network depicted in Figure 2.1 is suitable for a classification task with $K = 4$ classes with $\phi^{(2)} = softmax$. At the output layer, the probability of class k as predicted by the model corresponds to the k^{th} output neuron and is computed via Equation (2.5).

$$\hat{y}_k = \frac{e^{W_k^{(2)} \cdot h^{(1)}}}{\sum_{k'=1}^{K} e^{W_{k'}^{(2)} \cdot h^{(1)}}} \tag{2.5}$$

The class assigned the highest probability is the final output of the network.

$$\hat{k} = \arg\max_k \hat{y} \tag{2.6}$$

The flow of information through the network described so far is referred to as the *forward pass* of the network, and is performed to obtain the model prediction at test and train time. The *backward pass*, on the other hand, is performed only during training when the parameters of the network are being adjusted as follows.

When the predictions of a neural network are in the form of a probability distribution over the possible output classes, the network parameters can be optimized

with the use of the principle of *maximum likelihood* [Goodfellow et al., 2016]. The goal of maximum likelihood estimation is to discover the parameter values W that give the distribution which maximizes the probability of observing the training data under the model.

The objective during training a neural network is to minimize a cost function by adjusting the parameters of the model W. The cost function J can be defined as the negative log-likelihood, or the *cross-entropy* between the true distribution as defined by the training data and the predicted model distribution. Minimising the negative log-likelihood is equivalent to maximising the likelihood of the training data, while using the logarithm function ensures numerical stability of the computation.

Equation (2.7) summarizes the computation of the cost function for a training set with K output labels, where $\hat{y}$ contains the predicted probability distribution, and y contains the desired distribution.

$$J(W) = -\sum_{(x,y)\in S}\sum_{k=1}^{K} y_k \times log\Big(\hat{y}_k\Big) \tag{2.7}$$

Minimizing the cost function can be achieved through the use of the *gradient descent* algorithm, which finds the derivative of the cost function with respect to each of the network parameters, and then adjusts the weights in the direction of the negative slope. *Back propagation* is a technique which enables the efficient calculation of the gradient descent via repeated application of chain rule for partial derivatives [Graves, 2012].

MLPs have been successfully applied to a multitude of natural language processing problems. Furthermore, the availability of good quality distributed word representations has rendered complex feature engineering obsolete for many tasks. Instead, attention has been redirected towards the use of more complex neural models, better adept at handling the specificities of natural language.

One drawback of FFNNs with respect to their application to NLP tasks is that their predictions only depend on the current input to the network, but cannot be conditioned on any past or future inputs. When dealing with natural language this can be of a disadvantage due to its inherent sequential nature – tokens are produced

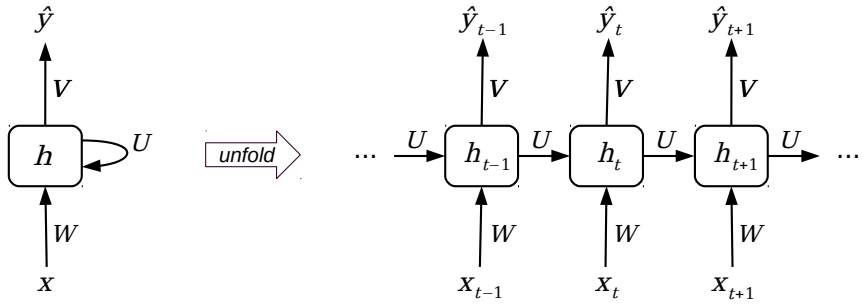

FIGURE 2.2: An example of a recurrent neural network with one hidden layer. To the left, a RNN in compact form, with a recurrent connection at the hidden layer. To the right, the unfolded computational graph of the same network over several time steps.

as parts of sentences, and sentences – as parts of larger discourse sequences, rather than in isolation. While for some NLP tasks it might be sufficient to consider n consecutive tokens as a way to introduce more contextual information when employing a FFNN, a fixed size window is not always easy to determine or suitable.

In the following sections we will turn to neural architectures suitable for dealing with sequential data.

2.2.3 Recurrent Neural Networks

Figure 2.2 provides an example of a Recurrent Neural Network (RNN) architecture with a single hidden layer. The depiction to the left illustrates the cyclic connection present in the network, namely an outgoing edge from the hidden layer pointing back to itself. The one to the right presents the computational graph of the same network when *unfolded* over a sequence of inputs.

The forward pass of an RNN is similar to that of an MLP with a single hidden layer, except that the hidden layer h_t is computed from the output of the hidden layer from one step back, h_{t-1}, in addition to the current external input vector x_t (Equation (2.8)).

$$h_t = \phi\Big(U \cdot h_{t-1} + W \cdot x_t\Big) \tag{2.8}$$

At each step the network may use the current hidden layer activations h_t to produce an output $\hat{y}$ via Equation (2.9). For instance, in POS tagging each vector x would

correspond to a real-valued representation of a token in the sentence, and each vector $\hat{y}$ would be a probability distribution over a set of POS tags. In other tasks, however, it might be beneficial for the entire input sequence to be read before a classification decision is made based on the final hidden layer state, h_n.

$$\hat{y}_t = \text{softmax}\left(V \cdot h_t\right) \tag{2.9}$$

There are two key distinctions between RNNs and FFNNs, namely the concept of internal memory and parameter sharing.

RNNs are said to have an *internal state* or *memory*. The recurrence in the network creates a mapping of an arbitrary length sequence to a fixed length vector h_t, which can be seen as a lossy summary of the past input up to time step t [Goodfellow et al., 2016]. Thus, the predictions of the network are based on a history of previous inputs preserved in the network's hidden state. In comparison, an MLP can base its decision only on the current input at any given time.

RNNs share parameters across time steps. The same parameters matrices W, U, and V, are involved when computing the network hidden state and output prediction at any given time step. The model always has the same input size and is not limited to sequences of specific lengths seen during training. The parameters can be thought of as defining the *transitions* of the network from one state to the next [Graves, 2012]. In addition, parameter sharing enables the network to generalize better from fewer training examples [Goodfellow et al., 2016].

Training an RNN is performed via a form of back-propagation referred to as Back-Propagation Through Time (BPTT). Similarly to the standard version of the algorithm, BPTT consists of repeated applications of the chain rule for partial derivatives. Due to the presence of a recurrent connection, however, the output produced at time step t depends on the whole sequence of hidden activations and inputs up to this point. The unfolded computational graph is utilised as an explicit description of which computations to perform when adjusting each of the network parameters.

For many tasks it could be beneficial to have access to future, as well as past contexts. When dealing with natural text, for instance, the full input sequence is usually available to the classifier in advance and a look-ahead could often prove

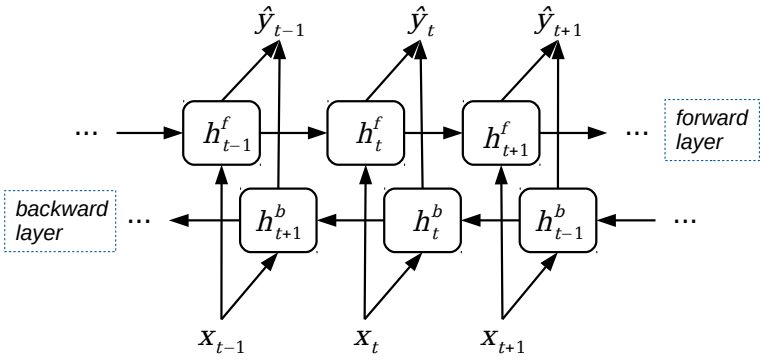

FIGURE 2.3: An example of a Bi-directional Recurrent Neural Network. The forward and backward RNN layer are denoted as h^f and h^b, respectively.

useful in guiding the classifier's prediction. Bi-directional Recurrent Neural Networks (Bi-RNNs) are an extension to standard RNNs which take advantage of the predictions of two recurrent layers: one which processes the input from beginning to end, and another which goes through it in reverse (Figure 2.3). The hidden layer activations of each layer are then concatenated (Equation (2.10)) to provide a symmetrical past and future context to the output layer of the network.

$$h_t^{\text{Bi-RNN}} = [h_t^f, h_t^b] \tag{2.10}$$

As discussed in the previous sections, the biggest advantage of RNNs over FFNNs is their ability to utilise contextual information when mapping between input and output sequences. In practice, however, the context stored in the network internal state is limited. The network's recurrent connections can result in very deep computational graphs for long input sequences, which renders them susceptible to the issues of *vanishing* or *exploding* gradient. The gradient refers to the magnitude and direction in which each network weight is updated during training. The computation of the gradient may involve repeated multiplication of very small or large values, thus leading to exponential growth or decay in its value as the distance between the weight being updated and the target output grows. When dealing with long range dependencies, the gradient becomes more likely to reach values very close to zero, thus preventing the network from further weight change and consequently learning, or to blow up, making learning unstable [Goodfellow et al., 2016]. The neural architecture discussed in Section 2.2.4 has been proposed as a way to counter these effects.

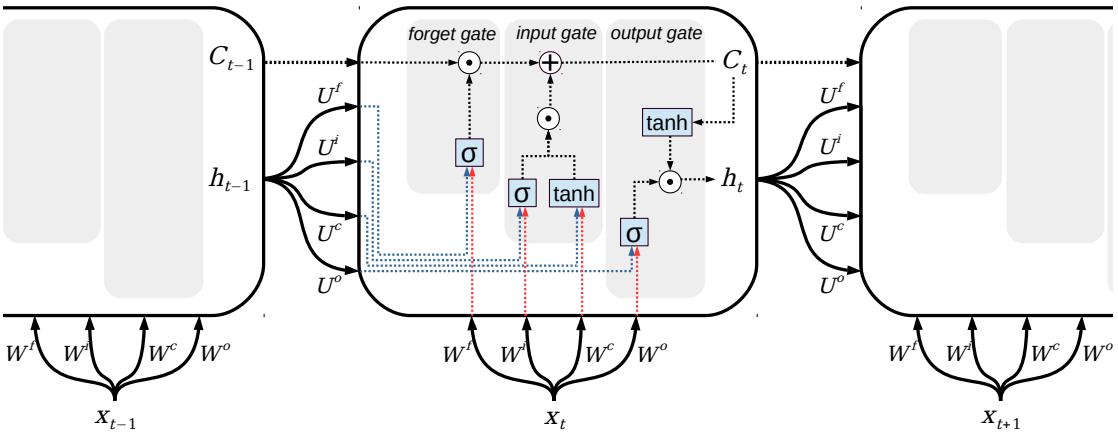

FIGURE 2.4: An example of an LSTM unit at time step t. The dotted lines represent the flow of information within the unit through the forget, input, and output gates, while the solid lines represent edges associated with network parameters. The symbols $\odot$ and $\oplus$ denote the operations element-wise multiplication and addition, respectively.

2.2.4 Long Short-term Memory Networks

Long short-term memory networks (LSTM) [Hochreiter and Schmidhuber, 1997] are a type of RNN architectures which aim to explicitly control for the information flow from and into the network's memory via a system of gating units.

The LSTM layer is composed of an LSTM unit which contains an internal recurrent memory *cell* C, in addition to the outer unit recurrence present in standard RNNs [Goodfellow et al., 2016]. The memory cell is intended to model the long-term memory of the network. At each time step, the gates of the unit decide which information to store, remove, or access from the cell. Figure 2.4 provides a schematic representation of the internal structure of an LSTM unit over a sequence of time steps.

The forward pass of an LSTM is similar to that of an RNN, with Equation (2.8) replaced by the computations summarised below:

$$f_t = \sigma(U^f \cdot h_{t-1} + W^f \cdot x_t) \tag{2.11}$$

$$i_t = \sigma(U^i \cdot h_{t-1} + W^i \cdot x_t) \tag{2.12}$$

$$C'_t = \tanh(U^c \cdot h_{t-1} + W^c \cdot x_t) \tag{2.13}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot C'_t \tag{2.14}$$

$$o_t = \sigma(U^o \cdot h_{t-1} + W^o \cdot x_t) \tag{2.15}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.16}$$

A typical LSTM unit contains a *forget* gate, an *input* gate, and an *output* gate, each of which has a similar basic structure. The behaviour of the gates is controlled by several parameters adjusted during training. Each gate consists of a layer with sigmoid activation σ, taking as input the hidden activation from the previous time step, h_{t-1}, and the current input, x_t.

The forget gate, f_t, is intended to learn to delete information from the memory cell via element-wise multiplication ($\odot$) with the sigmoid activation, which produces values $\in (0, 1)$. Some of the information stored in the cell may thus be "forgotten" when multiplied by values close 0, or retained otherwise. The input gate i_t is similar to the forget gate, but is intended to learn which new information from a candidate cell C' to add to the cell state. The updated cell state is obtained via Equation (2.14). Finally, the output gate, o_t, is intended to control for the extend to which certain information from the memory state is involved in computing the unit's hidden activation, h_t. The final hidden activation at time step t is obtained via Equation (2.16).

Similarly to the RNN architecture described in Section 2.2.3, the LSTM layer can be combined with an output layer to produce a prediction at each time step, or after reading the complete input sequence (Equation (2.9)). Furthermore, a Bi-directional Long Short-Term Memory network (Bi-LSTM) can be defined to take advantage of future, as well as past context (Figure 2.3).

Training an RNN is performed with the BPTT algorithm mentioned in Section 2.2.3. LSTM networks have been shown to be better adept at learning long-range dependencies than vanilla RNNs [Goodfellow et al., 2016] and are less susceptible to the vanishing gradient problem [Graves, 2012].

2.3 Distributed Word Representations

This section offers an overview of word embedding models. Word embeddings have become a standard way of representing natural language text in machine and deep learning models. In this work they are utilized in two ways. The Bi-LSTM event detection model discussed in Chapter 4 relies on contextualized word embeddings (see Section 2.3.3) for its input representation. They are central to the experiments described in Chapter 6, which investigate whether classic word embedding models such as `word2vec` and `glove` (Section 2.3.2) can serve as a source of semantic knowledge for the task of coreference resolution.

2.3.1 Definition

Word embeddings are real-valued vector representations of words. While the term is typically used to denote word representations obtained as a byproduct of training a probabilistic language model [Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013a,b], a long line of work in distributional semantics has been devoted to the construction of vector representations from word co-occurrence statistics [Erk, 2012; Turney and Pantel, 2010]. The former set of methods are also referred to as *predictive*, and the latter as *count-based* [Baroni et al., 2014]. What both of these approaches have in common is that they aim to approximate a word's semantics from its use in context. The idea can be summarized by the distributional hypothesis [Harris, 1954], stating that words which occur in similar contexts tend to have similar meaning. The relationship between a word's meaning and its distributional characteristics could thus be exploited in order to approximate the former from the latter. Moreover, the availability of large volumes of written text makes it possible to generalize over a multitude of contexts per word, without the need for costly manual annotation.

In this work, the focus lies on predictive models, which formulate the relationship between a word and its context as a prediction task with a language modelling objective. We distinguish between non-contextualized approaches which assign a single fixed word representation per token from the vocabulary (Section 2.3.2), and those which assign separate context-dependent representations to each instance of a word (Section 2.3.3).

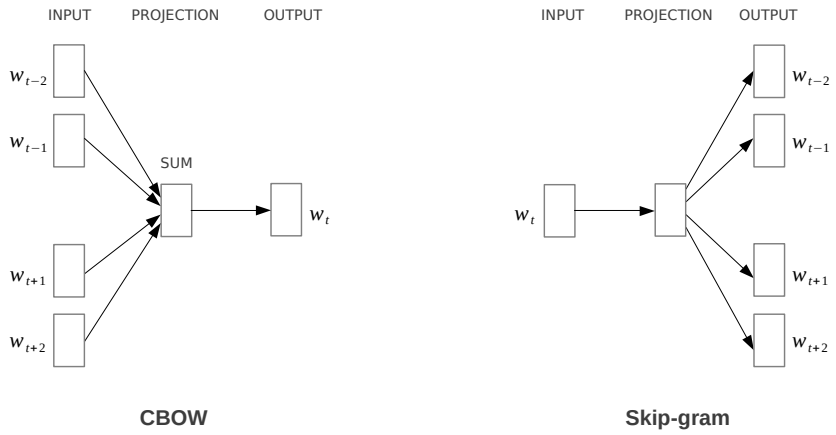

FIGURE 2.5: Continuous bag-of-words (CBOW) and skip-gram `word2vec` architectures [Mikolov et al., 2013a].

2.3.2 Non-contextualized Word Embeddings

One of the first predictive approaches proposed by Bengio et al. [2003] lays the foundation for the majority of the subsequent methods for learning distributed word representations [Collobert and Weston, 2008; Mikolov et al., 2013b,a]. Bengio et al. [2003] employ a single-layer FFNN n-gram language model. An n-gram language model breaks down the computation of the probability of a sequence of tokens, $P(w_1, .., w_n)$, into the product of the conditional probability of each token given its immediate preceding context of size n, $P(w_t|w_{t-n+1}, ..w_{t-1})$. At any given time step, the model aims to predict the most likely continuation of the n-gram sequence. Context words are fed into the network in the form of one-hot-vectors and projected to the hidden layer with matrix W, followed by a non-linear activation. In the output layer, a softmax is used to estimate a probability distribution over all tokens in the vocabulary, from which the most likely target word is predicted. The rows in the learned weight matrix W correspond to the word embeddings of the tokens in the vocabulary.

The `word2vec` model [Mikolov et al., 2013a,b] comes in two flavors which differ in the formulation of their language modelling training task, namely continuous bag of words (CBOW) and skip-gram (Figure 2.5). CBOW's training objective more closely resembles that of previous work, but allows the model to simultaneously consider both past and future contexts when predicting a word in the sequence. The context of token w_t is defined as a symmetrical window of c tokens preceding and following the target word, $w_{t-c}..w_{t-1}, w_{t+1}..w_{t+c}$. The training objective is to

maximize the average conditional log probability of each word given its context, as summarized in Equation (2.17).

$$J = \frac{1}{T}\sum_{t}^{T} logP(w_t|w_{t-c}..w_{t-1}, w_{t+1}..w_{t+c}) \tag{2.17}$$

The skip-gram version of the model focuses on the reversed scenario where the target word is used to predict the tokens from its context. The model aims to maximize the sum of the log probability of each context word given the target word, averaged over all words in the training data (Equation (2.18)).

$$J = \frac{1}{T}\sum_{t}^{T}\sum_{\substack{-c\leq j\leq c \\ j\neq 0}} logP(w_{t+j}|w_t) \tag{2.18}$$

A key to the success of word2vec is its ability to efficiently leverage large amounts of training data. This is accomplished via several modifications aimed at reducing the complexity of training a FFNN language model similar to the one employed in [Bengio et al., 2003]. For instance, the non-linearity of the hidden network layer is omitted, sacrificing some of the expressiveness of the model in favor of boosting its efficiency [Mikolov et al., 2013a]. The main computational bottleneck stems from the use of the softmax function at the output layer [Bengio et al., 2003]. At each training step, the softmax computes a probability distribution over all tokens in the vocabulary. When dealing with a large corpus, the vocabulary size can reach billions, rendering this operation extremely expensive. Mikolov et al. [2013a,b] propose two alternatives for circumventing the use of the softmax, namely *hierarchical softmax* and *negative sampling*.

Hierarchical softmax is a technique for approximating the softmax operation. The network output is organised into a binary tree, with leaf nodes corresponding to the tokens in the vocabulary. Rather than predicting a probability distribution over all possible outputs, the network performs a number of binary classifications which predicts a path from the root to the leaf of the tree corresponding to the desired output. A good choice for a binary tree structure is the *Huffman tree*, where shorter paths are assigned to more frequent tokens, and vice versa. The probability of a certain prediction can be reconstructed via the sigmoid outputs produced while traversing the path, but is not necessary for training the model.

Negative sampling is a slight reformulation of the original objective of the model. Rather than predicting an output word based on a given input word, the model predicts whether a pair of words are co-occurring or not. For each pair drawn from the corpus, a set of k negative examples are randomly *sampled* by selecting alternative tokens from the vocabulary to serve as context words. Thus, the model performs $k+1$ logistic regressions at each training iteration.

Mikolov et al. [2013b] introduce two further improvements to the word2vec models, namely *subsampling* and embedding of *phrases*. Subsampling addresses the word frequency imbalance in the training data. As highly frequent tokens such as determiners and prepositions co-occur with most other words, they do not contribute much valuable information to their word embeddings. Discarding very frequent words improves training speed and the quality of the word representations of less frequent words. The other modification of the model is aimed at improving the embeddings of multiword expressions. A simple data-driven approach is used to identify such expressions (e.g., New York) so that they can be treated as single units during training, accounting for their non-compositional meaning.

Levy and Goldberg [2014] propose an extension of the skip-gram word2vec model which utilizes non-linear contexts. Instead of including the words immediately preceding or following the target word in a window of a predefined size, the context is defined by following a path in the dependency parse tree from the target word. By picking out only the head and dependent tokens, this approach allows for the inclusion of relevant words which are further away from the target word and might have been otherwise missed by the linear context, as well as for the exclusion of less-relevant but closely situated tokens.

Levy and Goldberg [2014] argue that the resulting word representations encode different semantic information, exhibiting less-topical and more functional similarity of a co-hyponym nature. For example, the top 5 most similar words to "Florida" according to the original model were counties and cities within the area, while the syntactic context provided other state names. Levy and Goldberg [2014] summarize the distinction as words which "associate with the target word" and words which "behave like the target word". This tendency is further demonstrated in a quantitative evaluation on Agirre et al. [2009]'s split of the WordSim353 data set [Finkelstein et al., 2001], which differentiates between word relatedness (topical similarity), and similarity (functional similarity). This property of the syntactic word embeddings can be of interest for tasks such as named entity recognition and

coreference resolution, where such information is usually provided via fixed lists of entity names [Daumé III and Marcu, 2005].

Pennington et al. [2014] propose `glove`, or global vectors, an alternative approach for obtaining word representations which brings together ideas from count-based and predictive models. Similarly to word2vec, glove predicts the embedding of a given word by observing words from its local context. However, rather than focusing on a single word–context pair at a time during training, glove operates on a word-word co-occurrence matrix, which holds a *global* view of all of the contexts a given word has been observed in.

Let $X_{V\times V}$ be a co-occurrence matrix where X_{ij} contains the number of times a target word i was observed together with a context word j, and V denotes the vocabulary size. A word's context can be obtained by employing a symmetrical or asymmetrical sliding window of a fixed size over the corpus. The weight with which a context word contributes to the count stored in X_{ij} is determined by its distance from the target word, such that a distance of d words contributes a weight of $1/d$.

The probability of a word co-occurring with a context word is maximized by performing a log-bilinear regression with a weighted least-squares objective function (Equation (2.19)). The model aims to obtain word and context vectors w_i and $\tilde{w}_j$ such that the sum of their dot product and the bias terms b_i and b_j equals to the logarithm of the co-occurrence counts for that pair, X_{ij}. The weighting function f (Equation (2.20)) is intended to minimize the effect of very infrequently occurring noisy contexts, as well as very frequent co-occurrences such as those involving stop words.

$$J(W,\tilde{W}) = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \cdot \tilde{w}_j + b_i + b_j - logX_{ij})^2 \tag{2.19}$$

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{2.20}$$

The training process results in two sets of word embeddings, W and $\tilde{W}$, which in the case of a symmetric context are roughly equivalent, save for their initial

random initialization. Pennington et al. [2014] propose to use the sum of the two matrices to obtain the final vector representation of each token.

A shortcoming of the word representation methods discussed so far is their inability to adequately represent out-of-vocabulary words in downstream tasks. One commonly used heuristic is treating low-frequency words in the training data as unknown and replacing them with a reserved token (UNK) in order to obtain a default embedding for such cases. This strategy is less than ideal, as it assigns the same representation to unrelated low-frequency words. It is especially problematic for languages with rich morphology which have very large vocabularies and thus many word forms may not be observed during training.

Bojanowski et al. [2017] propose `fasttext`, an extension of the skip-gram word2vec model which enriches word representations with sub-word information, making the model better adept at handling out-of-vocabulary words. The proposed strategy assigns word representations to character n-grams, in addition to complete words. Words are broken down into character n-grams, with special symbols marking their beginning and end. During training, a word is used to predict each of its context words in the same way as in the original skip-gram model with negative sampling, but the representation of the word is the sum of each of its constituent character n-grams. Summing over the resulting character embeddings can be performed in an analogous way post training to generate word representations for out-of-vocabulary words.

2.3.3 Contextualized Word Embeddings

Classic word embedding approaches such as `word2vec` and `glove` provide a single *fixed* representation for each token in the vocabulary, which offers a conflated view of all contexts in which the token was observed. When the token in question is polysemous or a homonym, however, the resulting embedding can be thought of as representing the average or predominant sense of that token found in the data set. This is undesirable as it can lead to imprecise representations built from unrelated word contexts. For example, the context in which "crane" would be observed would vary drastically when the word is used to denote a type of bird and when it is referring to an industrial machine used for lifting. Furthermore, the representations of unrelated words (e.g., "trap" and "computer") which co-occur frequently with an ambiguous word ("mouse") would be pulled towards each other

in the semantic space [Camacho-Collados and Pilehvar, 2018]. All of this can in turn negatively impact the performance of downstream applications relying on semantic information.

A number of works have attempted to tackle this issue by leveraging word sense information. Such representations, referred to as *sense embeddings*, assign a separate embedding to the different senses of each word. Unsupervised methods perform clustering of word contexts to induce word senses [Van de Cruys et al., 2011; Liu et al., 2015]. Multilingual data can also be exploited to determine the semantic classes of words from their translations in another language [Šuster et al., 2016]. Existing knowledge resources such as WordNet [Miller, 1995] can also be leveraged to obtain the sense inventory[2]. A common shortfall of these approaches is the need to perform an additional word sense induction or disambiguation step in order to use the obtained sense embeddings in a downstream task.

More recently, another class of methods for obtaining word representations dubbed *contextualised* word embeddings has gained popularity and has lead to advancements in the state of the art for a number of NLP tasks. Contextualised word embeddings differ from the models discussed so far in that they represent each token as a function of the entire input sentence. In providing a separate, context-aware representation for every occurrence of a token, they are by design less susceptible to the issue of adequately representing polysemous words. The success of these approaches can be further attributed to their use of deep neural network architectures. Last but not least, they incorporate techniques for handling out-of-vocabulary words through modelling sub-word information, thus tackling another limitation of the majority of classic approaches.

Peters et al. [2018] present `elmo`, or embeddings from language models. As the name suggests, elmo belongs to the class of predictive approaches which obtain word representations jointly with training a language model. The architecture of elmo consists of multi-layer forward and backward LSTMs with residual connections between each layer. The input to the network are context-independent representation of each token obtained by convolutional filters over character n-grams followed by highway layers [Srivastava et al., 2015].

[2]For a comprehensive review of sense embedding methods we refer the reader to [Camacho-Collados and Pilehvar, 2018]

The recurrent architecture enables the prediction of a token from its context without the need to limit the decision to a local window of fixed size. Instead, the probability of word t_k can be conditioned on the entire input sequence, with the forward layers basing the prediction on $t_1, ..t_{k-1}$, and the backward layers on $t_{k+1}..t_n$. Equation (2.21) summarizes the training objective of the model. The parameters for the input embeddings (Θ_x) and softmax layer (Θ_s) are tied during training, while the forward and backward language model parameters ($\overrightarrow{\Theta}_{LSTM}$, $\overleftarrow{\Theta}_{LSTM}$) are kept separate.

$$J = \sum_{k=1}^{n} \Big(logP(t_k|t_1, ..t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + logP(t_k|t_{k+1}, ..t_n; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big) \tag{2.21}$$

Peters et al. [2018] employ a 2-layer LSTM in each direction, and concatenate the resulting hidden states for the two at each layer. More generally, given L layers, the model obtains the following representations for token t_k:

$$R_k = \{h_{k,j}^{LM} | j = 0, .., L\} \tag{2.22}$$

where $h_{k,0}^{LM}$ corresponds to the character-based input embedding of t_k, and $h_{k,j}^{LM} = [\overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM}]$ for $j > 0$.

The multi-layer token representation R_k is collapsed into a single vector for inclusion in downstream tasks. Rather than taking only the top-most layer, however, Peters et al. [2018] propose to learn a linear combination of all of the layers, determined by the specific task at hand. The motivation behind this is that different NLP tasks might benefit more from the levels of abstraction provided by certain network layers. Intrinsic evaluation has shown that lower-level representations are more suitable for syntactic tasks such as POS tagging, while higher-level ones are better for semantic tasks such as word sense disambiguation. The final `elmo` representation for token t_k is estimated via Equation (2.23), where s^{task} denotes the task-specific softmax-normalized weights for each layer, and y^{task} is a scalar parameter of the task model which allows for the scaling of the entire elmo vector.

$$\texttt{elmo}_k^{task} = y^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM} \tag{2.23}$$

`Bert` [Devlin et al., 2019] obtains contextualized word embeddings with the help of a transformer architecture. Transformers were first introduced in [Vaswani et al., 2017] for the task of machine translation. The original model consists of two parts: an *encoder*, which creates a representation of the source language sentence, and a *decoder*, which utilizes this representation together with any partially translated output to generate a sentence in the target language. Bert makes use of the encoder part of the architecture to create sentence representations, from which context-dependent token representations can be extracted, similarly to Elmo.

Central to the transformer model is the mechanism of *self-attention* [Vaswani et al., 2017]. At any given position in the sequence, self-attention allows the model to take into account (pay *attention* to) any relevant part of the whole sequence, regardless of the distance between the two. It can be thought of as enriching the input embedding of each token t_k with information from the whole sequence of tokens, $t_1..t_n$. Vaswani et al. [2017] describe attention as "a mapping between a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors". To illustrate the concepts, consider the example sequence "the bank of the river". If the target token is "bank", the attention mechanism might learn that the token "river" carries relevant information. The target token is the query (Q) against which all tokens (keys, K) in the sentence are evaluated for relevancy. Finally, a new representation of the target token is created which incorporates different degrees of information from every token (values, V) in the sequence.

The computation of self-attention is summarized in Equation (2.24) [Vaswani et al., 2017]. The query Q would be the input embedding of the token "bank" in the running example, while the set of keys and values would be the embeddings of all tokens in the sequence. The dot product between the query and the matrix of keys is used to estimate a weight for scoring the importance of each key to the query. Intuitively, the dot product between "bank" and "river" might be a high, as the input embeddings of the two would share similarities across multiple dimensions. The obtained weights are dividing by a scaling factor[3] and normalized using the

[3] $\sqrt{d_k}$, where d_k represents the dimension of the input embedding.

softmax function. The new representation of the token "bank" would be a weighted sum of all input embeddings (V).

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{Q \cdot K^T}{\sqrt{d_k}})V \tag{2.24}$$

The transformer architecture involves multiple applications of the self-attention mechanism, also called attention *head*. In multi-head attention, the queries, keys, and values are linearly projected several times with separate parameters. The output of each head is concatenated and fed into a final linear layer. Furthermore, multiple layers of multi-head attention are stacked together, so that the output of one layer serves as input to the next. For instance, the pre-trained model Bert_{large} contains 24 layers with 16-head attention, resulting in 340M parameters [Devlin et al., 2019].

The input sequence is broken down into sub-word units and embedded with WordPiece [Wu et al., 2016]. Two additional learned embeddings are added to these input embeddins, one which encodes the position of the token in the sequence, and one which indicates whether the token belongs to the first or second sentence in the sequence. These positional embeddings are helpful as the transformer has no notion of word order.

Bert is trained on two tasks, next sentence prediction and masked language model, also known as *Cloze* task. The former is a binary classification aiming to predict whether two sentences appear consecutively in the corpus. This training scenario is meant support the use of Bert in downstream tasks which rely on understanding the relationship between two sentences, such as question answering, or language inference. The masked language model predicts hidden tokens chosen at random. This scenario allows the model to be truly bidirectional, taking into account future and past contexts for the prediction. In comparison, Elmo obtains a pseudo-bidirectional prediction through a combination of separate left-to-right and right-to-left language models. The problem with applying a bidirectional architecture such as a Bi-LSTM directly to a standard conditional language modelling problem is that when predicting token t_k in the sequence, the decision would be based on the hidden state $h_{k-1} = [\overrightarrow{h}_{k-2}, \overleftarrow{h}_k]$ from the previous prediction step, where $\overleftarrow{h}_k$ has already seen token t_k. Thus, the correct prediction of the target word becomes trivial.

Once trained, Bert can be utilized in downstream tasks in two ways. It can be used to represent the tokens in a sentence as (static) features, similarly to other word embeddings discussed so far. Alternatively, the model can be used as a starting point for fine-tuning on a downstream task operating on the sentence or token level. In this scenario, the model is initialized with the parameters obtained during the initial training on unlabelled data, and all model parameters are fine-tuned using labeled data for the downstream task.

A number of works have proposed extensions and modifications to Bert, aiming to improve the model's performance and speed. XLNnet [Yang et al., 2019], roberta [Liu et al., 2019] and albert [Lan et al., 2020] introduce changes to the training methodology of the original model, as well as utilize more training data in the process. Distilbert [Sanh et al., 2019] offers an approximation of the original model with less parameters.

Chapter 3

Background and Related Work in Event Extraction

3.1 Events

Events can be thought of as occurrences which differ from material objects in their relationship to space and time: objects are said to exist, while events are said to occur or happen or take place [Hacker, 1982]; objects have rather clear spatial boundaries, and vague temporal boundaries, while events are considered to have vague spatial boundaries and crisp temporal boundaries [Casati and Varzi, 2015]. Henceforth we will focus on linguistic events, or the natural language descriptions of real-world events.

A long line of work in theoretical linguistics has focused on defining events and event ontologies. Lexical aspect (*Aktionsart*) enables the sub-categorization of events based on the temporal properties of the verbs and phrases used to describe them, such as whether they extend in time, involve a change, and have a natural endpoint [Mani et al., 2005].

In his classical work, Vendler [1957] proposes a four-way classification of verbs based on their aspectual features. He distinguishes between *states*, *activities*, *accomplishments*, and *achievements*. States can be thought of as being non-dynamic, extended in time or permanent ("He *knows* the answer", "She *is* smart"). Activities are also extended in time, but involve change. They have no natural endpoint or goal ("I *'m running* in the park"). Accomplishments and achievements, on the

"Rudolph Giuliani will wed his companion, Judith Nathan, on May 24 in the ex-mayor's old home."

- **TRIGGER:** *wed*
- **ARGUMENTS:**
  - **person:** *Rudolph Giuliani*
  - **person:** *his companion*
  - **time-within:** *May 24*
  - **place:** *the ex-mayor's old home*
- **PROPERTIES:**
  - **modality**: *asserted*
  - **polarity:** *positive*
  - **genericity:** *specific*
  - **tense:** *future*

FIGURE 3.1: ACE event mention of type *Life.Marry*

other hand, have an endpoint and entail an underlying process. Achievements are conceived of as instantaneous ("He *reached* the bus stop"), while accomplishments involve an extended processes ("She *painted* a picture"). The term *event* encompasses all non-*states*, but in some works it is used more narrowly to refer to events with a terminus or delimitation, i.e. accomplishments and achievements [Rosen, 1999]. Furthermore, event is a property of the verb together with its modifiers and arguments, whose characteristics contribute to the event type of the entire clause [Rosen, 1999].

3.2 Events in Corpora

The Automatic Content Extraction (ACE) 2005 [Walker et al., 2006] data set contains annotations for entities and entity coreference, relations, events, and event coreference. The corpus was utilized in the ACE challenges focusing on three general areas: Entity Detection and Tracking, Relation Detection and Characterization, and Event Detection and Characterization. The data set consists of 599 documents from newswire, broadcast news, broadcast conversation, web log, discussion forums, and conversational telephone speech.

According to the annotation guidelines, an event is "a specific occurrence involving participants", which "can frequently be described as a change of state". An event annotation in ACE includes the span of text encompassing the event *mention*, the word which most clearly expresses the event – the event *trigger*, as well as zero

or more event participants, or *arguments*, together with the role they play in the event. Figure 3.1 presents an example of a sentence containing an event mention and all of its annotated components and properties.

ACE is a closed-domain event corpus, focusing on a predefined event ontology of interest. It includes annotations for 8 general types of events with 33 sub-types. The frequency of events varies greatly across types, with a few predominant ones, such as *Conflict.Attack*, and a high number of infrequent ones, as depicted in Figure 3.2. For three of the event types there are less than ten examples [Chen et al., 2017], while the median number of positive examples per type is only 65 [Ferguson et al., 2018].

The majority of the event mentions in ACE are triggered by verbs (60%) [Chen et al., 2017]. Other possible event trigger types include nouns ("The *wedding* took place in the afternoon") and adjectives ("The happily *married* couple") derived from verbs. The trigger is usually a single token, with the exception of phrasal verbs ("take out", "turn down"), some named entities ("War World II") and other Multi-Word Expressions (MWEs) ("tie the knot", "air strike").

Event annotations include several properties indicating the speaker's attitude towards the veracity of the discussed event and whether it occurred or not. For example, an event's *polarity* is negative if the text states that the event did not take place, as in "They are **not** being *sued*" (nevertheless considered of type *Justice.Sue*). An event's *tense* can be future, as in "Many small businesses **will** have to *close down*"(*Business.End-Org*). Unless an event can be interpreted as a singular occurrence at a specific place and time, it is annotated as *generic*: "The group specialized in *transporting* illegal weapons." (*Movement.Transport*). If the mention refers to a believed or hypothetical event, this is reflected in the modality property field as *not asserted*: "**Rumors** of *arrests* circulated in Vancouver" (*Justice.Arrest-Jail*). Special care needs to be taken when handling the mentions exhibiting some of these properties for NLP tasks where the *factuality* of events plays a role. Event extraction (Section 3.3) typically leaves such fine-grained analysis for a later stage and focuses on detecting event mentions regardless of whether or not they are expressed as corresponding to real-world events which have taken place.

The TAC-KBP data set is a more recently introduced closed-domain event corpus utilized in the Text Analysis Conference (TAC) Knowledge Base Population (KBP)

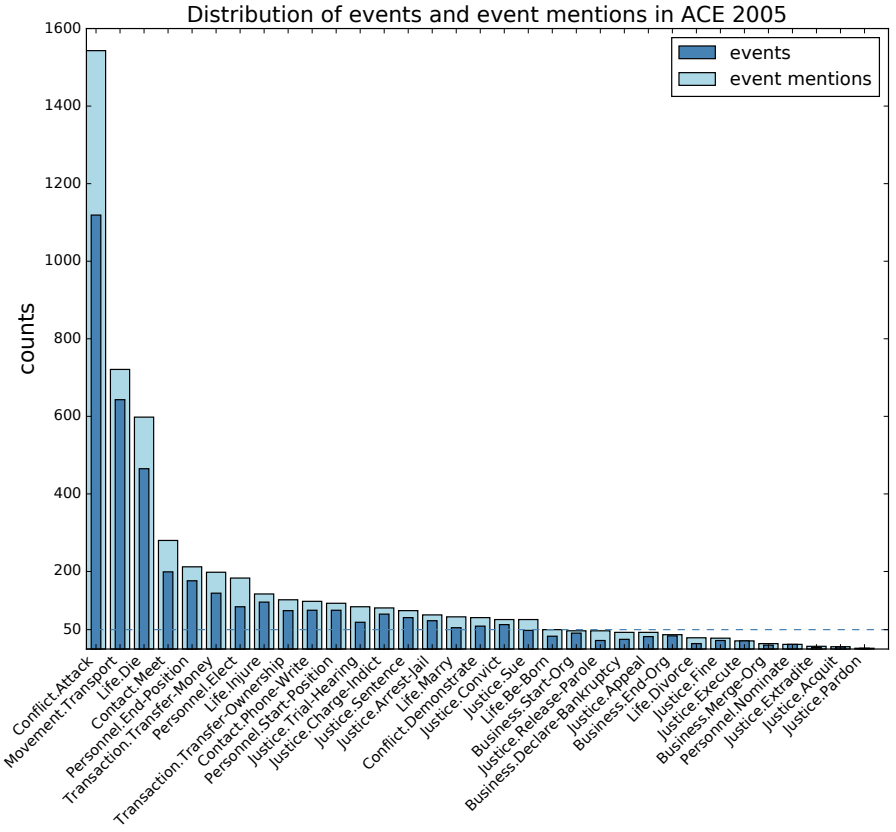

FIGURE 3.2: Distribution of events per type in ACE 2005, sorted by mention frequency.

tracks. The 2015[1] version of the corpus, consisting of 158 training documents and 202 test documents, is similar to the ACE 2005 data set in several respects. It includes annotations for 9 event types with 38 sub-types, which have a substantial overlap with those present in ACE. Furthermore, the source of the documents is similarly newswire and discussion forums. The main difference between the two data sets is that event triggers in TAC-KBP can be assigned more than one event type, referred to as double tagging [Mitamura et al., 2015]. For example, the trigger *murder* ("the *murder* of John") is assigned both a *Conflict.Attack* and a *Life.Die* label, as opposed to only *Life.Die* in ACE.

Two examples of open-domain event corpora include EventCorefBank (ECB) [Bejan and Harabagiu, 2010] and its extension EventCorefBank+ (ECB+) [Cybulska and Vossen, 2014], which contain within- and cross-document event and entity coreference annotations.

The corpus ProcessBank [Berant et al., 2014] offers entity and event annotations in the medical domain. The data set includes 200 paragraphs about biological

[1] https://tac.nist.gov/2015/KBP/Event/

processes, taken from a biology textbook. While no domain event types are annotated, the corpus includes information about event coreference and event causal relations.

3.3 Event Extraction

Information extraction (IE) is a broad term encompassing a number of natural language processing tasks aimed at the automatic extraction of structured information from text. It has gained popularity due to the growing amounts of information available in unstructured form. Relation Extraction and Event Extraction are two closely related IE tasks.

Event extraction deals with the detection of event mentions in text, as well as determining the entities participating in the event, if any. Events are typically viewed as occurrences involving participants which happen at a specific time and place. In the context of IE, it aims to answer *what* happened *where*, *when*, and *to whom*. Such information is valuable for a number of NLP tasks dealing with text understanding, such as question answering, and text summarization.

Event extraction encompasses several sub-tasks, namely trigger identification, trigger classification, argument identification, and argument role classification:

1. trigger identification – determine whether or not a word or a MWE triggers an event of interest.
2. trigger classification – determine the event type of the trigger (e.g., *Conflict.Attack*, *Life.Die*, etc.).
3. argument identification – determine which entities correspond to event participants.
4. argument role classification – assign role labels to the event participants (e.g., *Attacker*, *Target* of a *Conflict.Attack* event, etc.)

Trigger identification and classification are frequently performed in one step and are collectively referred to as *event detection* [Feng et al., 2016]. This task is challenging as the same event can be expressed through multiple different triggers,

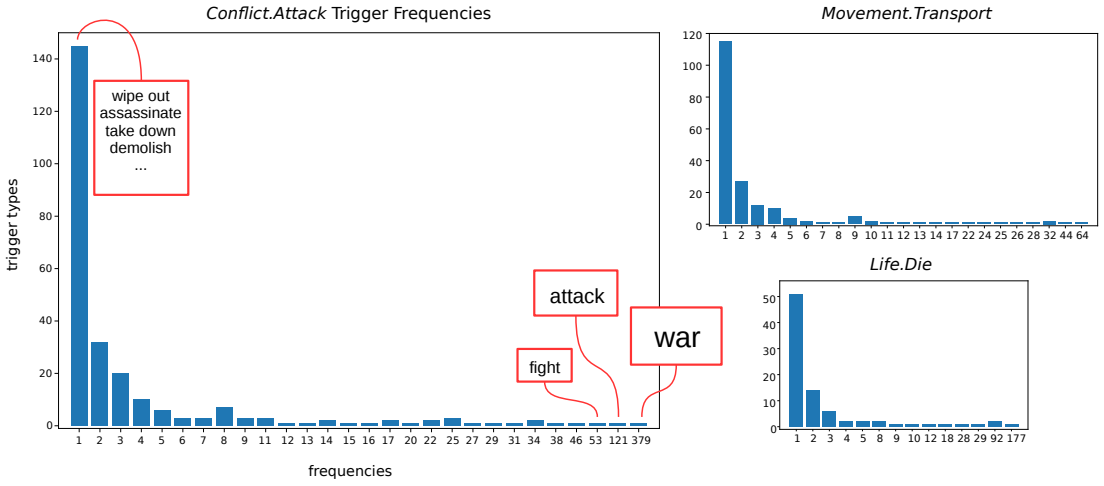

FIGURE 3.3: Event trigger frequency distribution for the three most frequent event types in ACE 2005. In all cases, there are a small number of predominant triggers (e.g., "war" appears 379 times as a trigger of *Conflict.Attack*), while the majority of trigger types appear only once.

and in some cases the same token can trigger several different events. For instance, the token "seize" can be used in the context of an attack event (*Conflict.Attack*), as well as to denote the change of ownership of a company (*Transaction.Transfer-Ownership*) or the change of a position of a person (*Personnel.Start-Position*). The frequency of trigger types varies in the training data, with (typically) a small number of frequently occurring triggers, and a large number of triggers which appear only once (Figure 3.3). Finally, an added challenge to the task is the detection of multiword triggers, where the correct span of the trigger needs to be determined as well.

Argument identification and classification are typically performed after event detection. A prerequisite for the tasks is determining the spans and types of entities (e.g., *Person*, *Location*, *Weapon*, *Vehicle*, etc.) present in the sentence[2]. The following steps include predicting whether an entity participates in a previously detected event, and what role it plays in that event. While a trigger can introduce a single event in a given context, an entity frequently participates in more than one event with different roles. For instance, in the following sentence, the cameraman is the *Victim* of a *Life.Die* event ("died"), as well as the *Target* of a *Conflict.Attack* event ("fired") [Nguyen et al., 2016]:

[2]Some of this information would be provided by a named entity tagger in a real-world scenario. As the process is error prone, authors often chose to use the gold-standard entity annotations available in ACE and focus on determining if an entity is an event argument.

> In Baghdad, a cameraman **died** when an American tank **fired** on the Palestine hotel.

Event Linking or *event coreference* is a subsequent task of event extraction which deals with grouping all mentions of the same event found in one or across several documents.

Relation Extraction involves discovering semantic relations between entities from text. The end goal of the task is typically the automatic knowledge base creation or population. A relation can be seen as an ordered tuple of entities or relation arguments. For instance, the relation indicating that two people are spouses can be represented as $spouse<\text{person}_1, \text{person}_2>$.

While event extraction can be seen as a sub-task of relation extraction, there are several differences between the two. A relation mention is defined by the presence of a minimal set of *obligatory* relation arguments [Krause, 2018]. Event mentions, on the other hand, may not involve any arguments beyond the event trigger, as is the case for approximately 15% of all mentions in ACE. Furthermore, relations represent *facts*. Event mentions can be non-asserted or negative (Section 3.2). Thus, only a subset of event mentions are of interest to relation extraction.

Nonetheless, there is a considerable overlap between the two tasks. Both face similar data sparsity issues, as they are highly domain specific – a corpus containing certain event or relation annotations cannot be employed to learn about any other type of event or relation. Thus, both tasks are often addressed by semi-supervised learning approaches.

The rest of this chapter is organised as follows: Section 3.3.1 presents the evaluation methodology employed in event extraction; Section 3.3.2 introduces the state-of-the-art approaches to supervised event extraction; Section 3.4 summarizes semi-supervised methods which have been employed for the extraction of additional labeled examples of events.

3.3.1 Evaluation

Each of the sub-tasks involved in event extraction is evaluated using the standard metrics precision (P), recall (R), and f-measure (F1), defined as follows:

$$P = \frac{true\ positives}{true\ positives + false\ positives} \tag{3.1}$$

$$R = \frac{true\ positives}{true\ positives + false\ negatives} \tag{3.2}$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.3}$$

We will use event detection as an example, but the argument role identification and classification are evaluated similarly.

An event is detected correctly (true positive) if and only if the correct token span of the event trigger, as well as the correct event type, were predicted by the model. False positives include cases where the model prediction of an event does not correspond to an event mention in the reference corpus. Finally, false negatives are event mentions missed by the model. The metrics precision and recall provide estimations of the correctness and coverage of the model's decisions, respectively. F1 score is defined as the harmonic mean between precision and recall.

There exist two variants of the f-measure, the *micro*- and *macro*-averaged F1 scores. In the former, precision and recall are computed for all predictions of the model, regardless of the type of event being evaluated. The latter is an average of the F1 scores computed separately for each event type (e.g., $F1_{Conflict.Attack}$, $F1_{Business.End\text{-}Org}$, etc.). In this setup every event type contributes equally to the final score, regardless of the number of mentions of that event present in the data set.

3.3.2 Supervised Event Extraction

A long line of work in event extraction has employed machine learning frameworks utilizing rich sets of features [Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013; Yang and Mitchell, 2016; Ferguson et al., 2016].

Features for the task can be broadly categorised as local or global. The former aim to model the local (within-sentence or smaller) contexts of triggers and arguments, while global features target the interdependencies between event types

within or across documents, or between event participants and event types in a larger context.

Examples of local features include token and POS tag n-grams, lemmas, NER types or entity types present in the sentence, synonyms, Brown clusters [Brown et al., 1992], head and dependent information extracted from a dependency analysis of the sentence, dependency path between an argument candidate and a trigger, and others [Li et al., 2013; Ferguson et al., 2016].

Global event features take a variety of forms. Ji and Grishman [2008] identity event triggers and arguments within each sentence and apply heuristic rules to ensure that the classifiers' predictions satisfy some document-wide event consistency constraints. Liao and Grishman [2010] note that local context is often not sufficient for resolving ambiguities when distinguishing between particular types of events. For instance, it can be challenging even for a human reader to determine if the sentence "He *left* the company" expresses that the person left their job (*Personnel.End-Position*), or physically left the company's grounds (*Movement.Transport*). In such cases, information from a wider scope, such as co-occurring event types, can serve to resolve the ambiguity. Hong et al. [2011] further note that entities of specific types frequently participate in specific events, and propose to use entity-type consistency as a feature to predict event mentions. Yang and Mitchell [2016] propose a joint inference model over entities, triggers, and arguments across a document to facilitate context-aware predictions.

Creating feature-rich event classifiers requires some expert knowledge and depends on the availability of a range of NLP tools and resources, which may not be applicable to many languages. Furthermore, the performance of such models can be negatively impacted by erroneous preprocessing.

The current state-of-the-art systems for event extraction employ neural architectures.

Nguyen et al. [2016] propose a joint framework for event trigger and argument identification and classification. Similarly to previous work employing a joint classifier [Li et al., 2013], they aim to model both local features relevant for the individual tasks, as well as global features to target the dependencies between them. An RNN architecture is employed in order to learn abstract sentence representations, which are in turn used together with additional features, to predict event triggers and argument roles.

The input to the model are sentences consisting of sequences of tokens w_1, .., w_n, as well as the gold-standard ACE entity mentions $e_1, .., e_k$, found in each sentence. After the model predicts the event trigger type of token w_i, it goes through each entity e_j present in the sentence and determines the role it plays in that event, if any.

The neural architecture employed to encode the sentence is a bi-directional Gated Recurrent Unit (GRU). The input to the network at a specific time step consists of the word embedding of the token, a real-valued embedding of the entity type, and a binary vector encoding the edges of the token in the dependency parse of the sentence.

The task inter-dependencies are modeled via memory vectors and matrices which store the predictions during the course of labeling the sentences so that past decisions about triggers and argument roles can be directly used in each prediction of the model. These include: vector G_i^{trg}, which indicates which event types have been recognized before time step i; matrix G_i^{arg}, which summarizes the argument roles that the entity mentions have played with some event in the past, and matrix $G_i^{arg/trg}$, which specifies which entity mentions have been identified as arguments for which event types so far.

The input for performing trigger classification of token w_i includes: h_i, local features consisting of a concatenation of the word embeddings of neighboring tokens, as well as the memory vector G_i^{trg}. A softmax layer is employed to obtain the final classification decision. The input for argument classification of entity e_j given trigger w_i includes: h_i and h_j, the local contexts of the two, as well as the two memory matrices, G_i^{arg} and $G_i^{arg/trg}$. Similarly, the argument role is obtained via a softmax layer. The network is trained by minimizing a joint negative log-likelihood function for triggers and argument roles.

The evaluation reveals that $G_i^{arg/trg}$ improves the performance of argument labeling, while the dependencies between trigger types G_i^{trg} and argument roles G_i^{arg} are not strong enough to be helpful for the joint model. Furthermore, when evaluated only on sentences containing more than one event, the proposed approach outperformed all the other related methods with large margins.

Feng et al. [2016] focus on the task of event detection. A hybrid neural network consisting of a Bi-LSTM and a Convolutional Neural Network (CNN) is employed to learn representations of each word in the sentence and predict its event type.

The Bi-LSTM is meant to encode the semantics of each word with its preceding and following context, while the CNN is meant to capture structural information from the local contexts. Unlike [Nguyen et al., 2016], no hand-crafted features or additional information besides the word embedding of every token are employed in the process.

Feng et al. [2016] use two convolutional filters (widths of 2 and 3) to produce local context representations from the sentence, combined with max pooling layers to obtain output vectors with fixed size. The final representation of each token is a concatenation of the forward and backward Bi-LSTM layers, and the two CNN output vectors. This combined representation serves as input to a softmax layer, producing the final classification decision.

This work has reported the best results for the tasks trigger identification and classification on the English portion of ACE 2005. Unfortunately several key details about the network parameters and training setup are omitted, which makes the exact reproduction of the result challenging.

3.4 Semi-supervised Approaches to Data Augmentation for Event Extraction

Labeled training data for event extraction and linking are limited in size and coverage of event types. Bootstrapping, distant supervision, and related methods have been employed to generate additional training data [Liao and Grishman, 2011; Chen et al., 2017; Ferguson et al., 2018].

Ferguson et al. [2018] extract clusters of coreferring event mentions from newswire articles. A major difference between this and other related approaches is that no knowledge base is employed in the process. Instead, simply the presence of matching entities in multiple articles published on the same date is considered an indicator that the articles discuss the same event. Their approach can be summarized in the following steps: (1) clustering of articles, (2) labeling of clusters, and (3) event trigger identification.

The goal of the first step is to identify and groups articles containing coreferring event mentions by finding rare entities mentioned frequently on a specific date. Pairs of articles are scored based on this intuition, and candidate clusters formed.

(1) *LSU* **fires** head coach *Les Miles* after 12 seasons.

(2) On Sunday morning, *LSU* athletic director Joe Alleva told *Les Miles* that the coach would **no longer represent** Louisiana State.

FIGURE 3.4: An example of two coreferring event mentions of different complexities from [Ferguson et al., 2018]

A drawback of this approach is that the position of events within the document is not taken into account. Thus, mentions involving entities participating in multiple events around the same time period (e.g. in elections) can be incorrectly assigned into the same cluster.

The resulting clusters are usually composed of mentions of different complexity (Figure 3.4). Some mentions which use a simpler language can be easily labeled with a supervised system. In order to predict the event type of each cluster, Ferguson et al. [2018] train a supervised system on the available labeled training data. A label is assigned to a cluster if it contains at least n events of a specific type.

Since existing supervised event extraction systems require event trigger annotations for training, the last step comprises the identification of the most likely trigger word for each sentence. The trigger is determined by computing the similarity between the embedding of each token from an event mention and the average of the embeddings of all triggers for this event type from the labeled data.

To evaluate the usefulness of the approach, Ferguson et al. [2018] augment two gold-standard event data sets, ACE 2005 and TAC-KBP 2015, with different amounts of the automatically extracted data, and use them to train a supervised event extraction system. They select a uniform number of examples per event type for the 19 most frequent event types in ACE. A micro-averaged F1 score over all events is reported for the identification of event triggers, computed using test samples from the original gold data sets. Event argument identification is not carried out and evaluated. The evaluation shows an improvement of 1.1 (TAC-KBP) and 1.3 (ACE 05) F1 score, compared to the respective baselines. The error analysis of 100 bootstrapped examples reveals that errors occurred mainly in steps (2) and (3).

Liao and Grishman [2011] present a bootstrapping approach to event extraction based on self-training which can be used to enrich existing labeled data sets for the task. Similarly to [Ferguson et al., 2018], this method relies on the intuition that event mentions found in a cluster of related documents can be tagged with higher confidence.

At the core of the extraction process are several maximum entropy-based classifiers for argument identification, argument role classification, and trigger classification, initially trained on the ACE'05 data set. These classifiers extract and label only high confidence mentions, which are in turn appended to the existing data, resulting in a larger corpus for retraining. This procedure is repeated several times.

Furthermore, two strategies are employed to ensure the quality of the extracted mentions: (1) extraction from clusters of related documents, to minimize error propagation during bootstrapping, and (2) cross-document inference, to encourage the extraction of examples of higher variability. An information retrieval system is employed in order to identify texts related to the ones already present in the gold data. Gold event arguments (names and nominals), together with their corresponding triggers, serve as queries for this system. The top n retrieved documents are selected to construct the document clusters used as input to the classifiers.

Liao and Grishman [2011] observe that the newly discovered examples tend to resemble the majority of examples already present in the labeled data set, and little novelty is introduced during bootstrapping. A likely explanation for this is that exactly these predominant gold examples are responsible for the high confidence classifier decisions. The issue is addressed via a global inference process which collects information about triggers and arguments from all coreferent event mentions in a document cluster, as opposed to a single one. Within- and cross-document entity coreference are employed in addition to expand the argument candidate information. In this way, a trigger or an argument can be added to the training data, even if its local context confidence is low, which in turn can result in a more varied data set.

Chen et al. [2017] utilise world and linguistic knowledge for the automatic labeling of event mentions. The proposed distant-supervision approach enables the extraction of event triggers, as well as event arguments, and is inspired by methods employed in relation extraction.

The source of world knowledge is the collaborative knowledge base *Freebase* [Bollacker et al., 2008]. It contains n-tuples of entities participating in various relations. The following is an example entry from the *people.marriage* category:

> <**spouse:** *Barack Obama*, **spouse:** *Michelle Obama*, **ceremony location:** *Trinity Church of Christ*, **from date:** *10/03/1992*, **to date:** –>

Finding text snippets in which these arguments co-occur is a reliable indicator of the presence of a marriage event mention. However, an inspection of the distribution of arguments in Wikipedia revealed that the cases in which all arguments of a Freebase event can be found in the same sentence are only 2% [Chen et al., 2017]. The knowledge base is therefore first used to select *representative* argument roles for each event type. In the above example, the two spouses are treated as such arguments for the marriage event, while the date and ceremony location are considered optional. The list of key arguments is employed to detect sentences containing candidate event mentions in Wikipedia text for the purpose of trigger extraction.

The next step involves the generation of trigger word lists for each event type. The strength of association between each verb from the previously extracted sentences and their corresponding Freebase event type is measured in order to select an initial list of verb trigger candidates. Since the list is noisy, and only contains verb triggers, FrameNet is employed to further filter the candidates and extend the list with noun triggers. For this purpose, a mapping is established between Freebase events and FrameNet frames with the help of a word embedding-based similarity score.

Finally, sentences containing at least one Freebase argument and a trigger from the filtered trigger list are extracted from Wikipedia via distant supervision. This results in over 70K automatically labeled examples for 21 Freebase event types, with the largest being *People.Marriage* (over 26K sentences).

Very high precision is reported for trigger labeling (88.9) and argument labeling (85.4), according to a manual evaluation of the extracted examples. An additional automatic evaluation is carried out by augmenting ACE events with corresponding automatically extracted examples, such as *People.Marriage* and ACE's *Life.Marry*. This data is used to train a CNN event extraction system [Chen et al., 2015], while a gold portion of ACE is reserved for evaluation.

The evaluation reveals an improvement of 0.8 F1 score for trigger identification, 1.4 F1 increase for trigger identification and classification, 4.2 for argument identification, and 2.2 F1 improvement in argument role identification, compared to the same system trained on ACE data exclusively. Furthermore, a model trained only using the automatically extracted and labeled data achieves a competitive performance.

The lack of sufficient amount of training data is often singled out as the main reason for the low accuracy of supervised event extraction systems. Both Chen et al. [2017] and Ferguson et al. [2018] present approaches for the automatic extraction of large quantities of additional data for the task. For instance, the number of *People.Marriage* event mentions extracted by Chen et al. [2017] exceeds 26K, while the corresponding ACE event *Life.Marry* contains only 83 mentions. However, despite the impressive size of the extended data sets, the performance of event extraction systems trained on them does not exceed that of their corresponding baseline models by a large margin.

A possible explanation for this is that the newly acquired data is not complementary enough to the event annotations in existing corpora. Since the evaluation is always carried out on a gold sample of ACE, the additional data has to target enhancing the existing data in terms of variety of training examples, and not merely quantity. The additional training data, despite being of good quality, might fail to provide enough examples of specific cases, problematic for event extraction systems.

Unfortunately, both works lack sufficient error analysis to support or refute this hypothesis. There is insufficient discussion of the kinds of examples present in the extended data sets as well. It would be interesting to know what the distribution of events in terms of number and type of arguments is, compared to the original data. The approach proposed by Liao and Grishman [2011] is especially interesting because it identifies and tries to tackle the lack of variability in automatically extracted corpora, an issue rarely discussed in related literature.

One evident difference between the automatically generated data and ACE is in terms of number of event arguments. 15% of all ACE mentions contain zero arguments, and this varies depending on the type of event. For instance, out of the 1543 mentions of *Conflict.Attack* events, 423 contain no arguments at all, as is the case with 25 out of the 83 *Life.Marry* mentions. The automatic detection

of these mentions can prove more problematic for existing systems compared to those with multiple arguments. Still, all of the aforementioned approaches for data generation are argument-based and cannot extract such examples. Having a large quantity of examples with multiple arguments in the augmented training data leads to an imbalance, which may further diminish the importance of detecting zero argument mentions when training an automatic event extraction system.

Chapter 4

A Self-training Approach to Data Augmentation for Event Extraction

4.1 Introduction

The state-of-the-art in event extraction (Section 3.3.2) are neural approaches which rely on the availability of sufficient amounts of annotated event data. Existing corpora for the task are limited in terms of size and coverage of event types. The most widely-used corpus, the ACE 2005 data set, has been shown to exhibit a number of data sparsity issues (Section 3.2).

The lack of sufficient amount of training data and the cost associated with building human-annotated corpora have inspired several works to investigate methods for the automatic extraction of additional labeled examples (Section 3.4). A variety of approaches including bootstrapping, distant supervision, and related methods, have been employed. These rely to different extends on the availability of some resources for the task, such as corpora or knowledge bases. Despite the fact that several of these works manage to extract large quantities of additional examples, the performance of EE systems trained on the extended data sets typically does not exceed that of their corresponding baseline models by a large margin.

The work presented in this chapter investigates an approach for the automatic extraction of additional training examples via self-training. We start off with a Bi-LSTM event tagger trained on an existing gold standard event corpus. This tagger is used to extract new labeled examples from a large collection of text. We

He	was	**swept**	**out**	**of**	**power**	after	a	popular	revolt	in	October	.
O	*O*	*B*-PE	*I*-PE	*I*-PE	*I*-PE	*O*	*O*	*O*	*O*	*O*	*O*	*O*

FIGURE 4.1: BIO tag representation of a sentence containing the multi-word event trigger "swept out of power" (PE:*Personnel.End-Position*).

employ a confidence filtering strategy to encourage the extraction of good quality examples. The resulting data set is in turn used to augment the existing gold corpus in a number of ways for re-training.

4.2 Task Definition

We focus on the task of event detection, which involves detecting event triggers and classifying them into one of several event categories. In the case of the ACE corpus employed in the current work, these include the 33 event types listed in Table 4.1.

What is understood under an event *trigger* is the word which most clearly expresses the event in the sentence, with the condition that a single trigger may not denote more than one event in a given context. However, a trigger word may be a multi-word expression, which is the case for approximately 14% of all trigger types in ACE. For this reason we have chosen to mark the spans of event triggers with begin-inside-outside (BIO) tags and view the task as a sequence labelling problem.

Figure 4.1 presents an example BIO encoding of a sentence. The first token of the trigger "swept out of power" is marked with the begin (B) tag, and each consecutive token with the inside (I) tag corresponding to the event type of the trigger. Every token which is not part of an event trigger is marked with the outside (O) tag. Event detection involves correctly determining of the full span of the trigger "*swept out of power*", as well as predicting that it introduces a *Personnel.End-Position* event.

Given a sequence of tokens w_1, .., w_N, we would like to predict the corresponding sequence of event tags, t_1, .., t_N, which encodes the correct event trigger spans and types, where N equals to the number of tokens in the input sentence.

4.3 Event Detection Model

The event detection model employed in this work is a Bi-LSTM sequence tagger [Plank et al., 2016; Plank and Agić, 2018; Ma and Hovy, 2016].

Throughout this section we will use the notation x and y to refer to the network input and output, respectively. Vector $x = (x_1, .., x_N)$ corresponds to the real-valued representations of the tokens in the input sentence, while vector $y = (y_1, .., y_K)$ corresponds to the numerical representation of the set of K possible event tags. In the current experiments, $K = 67$ classes, which include the B- and I- tags for each ACE event type and the O tag for non-triggers.

The Bi-LSTM network produces two hidden representations of each sentence, by traversing the input sequence x from beginning to end and in reverse. The representation of a token at time step i, h_i, is obtained by concatenating the forward and backward hidden layers of the network up to this point, and thus incorporates both the preceding and following sentential context of the token:

$$h_i^{\text{forward}} = \text{LSTM}(h_{i-1}^{\text{forward}}, x_i) \tag{4.1}$$

$$h_i^{\text{backward}} = \text{LSTM}(h_{i+1}^{\text{backward}}, x_i) \tag{4.2}$$

$$h_i = [h_i^{\text{forward}}, h_i^{\text{backward}}] \tag{4.3}$$

where LSTM denotes the computations performed at each LSTM cell (see Equations (2.11) to (2.16)).

Finally, the network predicts a probability distribution over the different BIO event tags. For each position i this involves computing the K-dimensional vector y_i.

$$y_i = \text{softmax}(V \cdot h_i + b) \tag{4.4}$$

where V is a parameter matrix, and b is a bias term.

The value y_{ik} can be interpreted as the probability of predicting event tag k given the input sequence x, or $P(t_i = k \mid x)$.

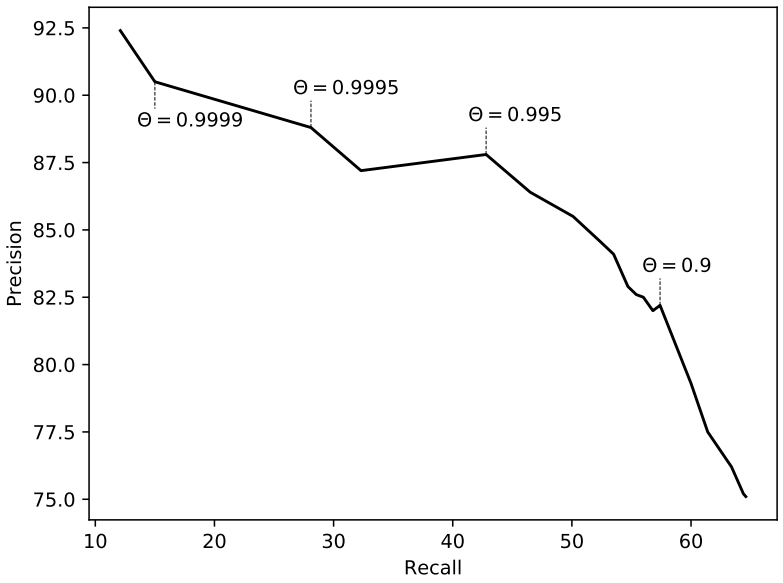

FIGURE 4.2: Precision-Recall trade-off for different values of the confidence parameter Θ, computed on the ACE development set.

4.4 Augmentation Data Extraction

We employ a self-training approach in order to extract additional labeled data for trigger identification and classification. Starting off with our baseline model, originally trained on the limited amount of available data in ACE, we label a large text collection. A confidence filtering strategy is utilised to retain only the most confident model decisions. The newly labeled examples are in turn used to augment the existing gold standard corpus, resulting in a larger corpus for re-training the model.

The approach to data augmentation for event detection proposed in the current work most closely resembles that of Liao and Grishman [2011] in its self-training nature. However, we perform re-training only once rather than in several iterations, motivated by the fact that re-training in multiple iterations can result in error propagation and amplification [Krause et al., 2012a]. In addition, we employ a *balancing* strategy (Section 2.1.2) in order to ensure that the number of new instances per class remains uniform, which has been shown to be beneficial in multi-class classification scenarios [Samad Zadeh Kaljahi, 2010]. Our data augmentation approach does not rely on any external resources for the task, besides

the data which is already available in ACE, pre-trained word embeddings, and a large collection of unlabelled data.

We run our event tagger on a subset of 5 million sentences from the Gigaword corpus [Parker et al., 2011], and apply a filtering strategy on the system's decisions in order to minimize errors as much as possible. To do so, we retain only those sentences in which all words were tagged with high confidence, i.e. where for some fixed Θ we can find BIO tags $k_1, \ldots, k_N$ such that $P(t_i = k_i \mid x) \geq \Theta$ for all i.

The Θ confidence threshold offers a way to boost the precision of the model at the expense of its recall. Figure 4.2 presents the effect of several values of the parameter on the performance of the tagger when evaluated on the ACE development set.

Using this strategy we obtained a large collection of new examples from the unannotated data. Table 4.1 presents the distribution of event types for different values of the filtering threshold in the resulting data set. Note that if a sentence contains a rejected tagger decision, it is discarded from the data set for this specific Θ value, as opposed to replacing the predicted tag with an O-tag. In doing so we aim to avoid the introduction of a possible false negative.

High Θ values lead to preserving only the most confident model decisions, which generally correspond to the most frequently occurring event types in the original ACE training data, but often result in the complete filtering out of less-frequent event examples. For instance, no *Justice.Pardon* events were tagged with confidence exceeding 0.995, and only 2 examples reached $\Theta \geq 0.9$. In order to preserve all event types in our augmentation data set, we utilize a dynamic Θ strategy: starting off with the most confident decisions of the model, we back off to a lower Θ until we reach the desired number of examples per event type.

The values of the variable Θ threshold we employed equal those depicted in Table 4.1, namely 0.9995 (most confident decisions), 0.995, 0.9, and 0.0 (no confidence filtering).

For our final augmentation data set, we selected a subset of the newly labeled examples of up to one thousand mentions per event type. This is a form of the *throttling* strategy mentioned in Section 2.1.2. We set on selecting a subset of the data in order to avoid overpowering the gold standard annotations during re-training.

Event Type	$count_{\text{ACE}}$	$\Theta = 0.0$	$\Theta = 0.9$	$\Theta = 0.995$	$\Theta = 0.9995$
Business.Declare-Bankruptcy	40	3378	2181	1159	430
Business.End-Org	31	1883	291	14	1
Business.Merge-Org	14	2270	900	0	0
Business.Start-Org	29	13245	4947	578	50
Conflict.Attack	1273	402590	257956	121948	49255
Conflict.Demonstrate	65	42125	25089	10089	2830
Contact.Meet	200	145310	97935	47215	18201
Contact.Phone-Write	112	11618	3964	484	64
Justice.Acquit	5	577	22	0	0
Justice.Appeal	30	5710	3420	1362	487
Justice.Arrest-Jail	78	36698	23556	12219	5884
Justice.Charge-Indict	96	24827	12507	4444	1591
Justice.Convict	64	7046	3389	1163	133
Justice.Execute	14	3028	1361	151	0
Justice.Extradite	6	343	3	0	0
Justice.Fine	22	3703	2047	609	71
Justice.Pardon	2	96	2	0	0
Justice.Release-Parole	46	7364	3426	570	9
Justice.Sentence	84	11682	6158	1832	607
Justice.Sue	60	7194	5028	2816	1045
Justice.Trial-Hearing	103	19801	12070	5730	2632
Life.Be-Born	47	5130	3334	940	110
Life.Die	524	138966	90475	43491	17368
Life.Divorce	20	830	610	208	13
Life.Injure	127	44808	26979	10474	2181
Life.Marry	73	4930	3514	2043	925
Movement.Transport	611	200880	115211	46537	17487
Personnel.Elect	162	40411	22420	8217	2469
Personnel.End-Position	159	38154	19331	7463	2897
Personnel.Nominate	11	2249	712	65	0
Personnel.Start-Position	92	26976	10487	2571	572
Transaction.Transfer-Money	128	18285	6911	1352	235
Transaction.Transfer-Ownership	92	14639	6579	1713	396

TABLE 4.1: Distribution of event mentions per event type in the ACE 2005 training set ($count_{\text{ACE}}$), and in our automatically labeled data set for different values of the confidence threshold Θ.

To avoid influencing too much the likelihood of an event occurring in a sentence, we also add a thousand negative examples consisting of sentences which do not contain any detected event mentions. Due to the abundance of such sentences in our automatically labeled data, these are filtered with the highest Θ threshold to avoid false negatives as best as possible.

4.5 Augmentation Approach

We experimented with several strategies for enhancing the existing training data with the newly extracted examples:

- AUG$_{concat}$: the augmentation data is appended to the training set.
- AUG$_{uniform}$: the loss when training on the augmentation data set ($\mathcal{L}_{\text{AUG}}$) is computed separately from the loss when training on the gold corpus ($\mathcal{L}_{\text{ACE}}$). A loss scaling constant (λ) is applied to $\mathcal{L}_{\text{AUG}}$. The total loss is computed as:

$$\mathcal{L}_{\text{TOTAL}} = \mathcal{L}_{\text{ACE}} + \lambda \cdot \mathcal{L}_{\text{AUG}} \tag{4.5}$$

- AUG$_{weight}$: similar to the above setup, but the scaling constant varies per event type e, proportionally to the frequency of the event in the ACE training set, $count_{\text{ACE}}(e)$[1]:

$$freq_e = \max(25, count_{\text{ACE}}(e)) \tag{4.6}$$

$$\lambda_e = \frac{freq_e}{10000} \tag{4.7}$$

- AUG$_{rev.weight}$: similarly to the above setup, the scaling constant varies per event type. However, the magnitude of weights is reversed so that lower weights are assigned to the more frequent ACE event types and vice versa:

$$\begin{aligned} freq_e = &\max(count_{\text{ACE}}(\hat{e})) - \\ &count_{\text{ACE}}(e) + 25 \end{aligned} \tag{4.8}$$

$$\lambda_e = \frac{freq_e}{10000} \tag{4.9}$$

In the first augmentation approach, AUG$_{\text{CONCAT}}$, the newly extracted examples receive the same importance as the original training set, while the remaining three strategies are meant to lessen the effect of the augmentation data during re-training. In the uniform scaling parameter augmentation (AUG$_{\text{UNIFORM}}$), we employ a scaling constant equal to 0.1 when training on the augmentation data set.

[1]We assume that each event was observed at least 25 times, to account for very infrequent events such as *Justice.Pardon*.

In $\text{AUG}_{\text{WEIGHT}}$, the loss while training on a batch predominantly containing an event of a certain type e is scaled by the frequency of that event in the training set divided by a constant. In this setup, frequently occurring event types are impacted more by their corresponding augmentation examples, with the intuition that, due to their higher frequency, they contributed to better quality augmentation data. In $\text{AUG}_{\text{REV.WEIGHT}}$ we aim for the opposite effect: frequent events receive little help from the augmentation data, as they already are well represented in the original training set, while infrequent event types receive a higher augmentation boost during training.

4.6 Experimental Setup

This section provides more details on the setup of the re-training experiments performed in order to evaluate the data extraction and augmentation approaches proposed in this chapter. Section 4.6.1 discusses some data choices beyond the ACE corpus. Section 4.6.2 presents some details about the evaluation setup of our experiments. Section 4.6.3 provides details on the implementation and training of our event tagger.

4.6.1 Data

The input to the Bi-LSTM tagger are sentences encoded with `Elmo` contextualized word embeddings [Peters et al., 2018] (Section 2.3.3). Elmo embeddings are obtained via a Bi-LSTM network trained with a language modelling objective, and assign a representation to each token which is a function of its sentential context. Thus, unlike classic distributed representations, they are not limited to providing a single embedding vector per token expressing its average or predominant sense. We expect this to be helpful for the current task, as trigger words can be ambiguous with respect to different event types (Section 3.3). Contextualized word embeddings better model such subtle variations of the word sense, and could thus provide valuable information to the tagger.

In order to investigate the effect of the domain of the augmentation data, we explored several sources of unlabelled examples, including a web crawl of RSS feeds [Ferguson et al., 2018], WaCkypedia [Baroni et al., 2009], and Gigaword.

The latter most closely resembles the domain of the ACE corpus. While all three led to comparable results, the Gigaword model was slightly better, in line with a previous finding where the best self-training gains were achieved when the domains of the unlabeled corpus and the seed corpus matched [Reichart and Rappoport, 2007]. Unless indicated otherwise, the results reported in the following chapters were obtained with augmentation examples originating from Gigaword.

In order to establish a comparison with a related approach for the extraction of labeled examples, we performed an additional experiment in which our augmentation strategy is used in combination with a different augmentation data set. The data, provided to us by Ferguson et al. [2018], was obtained through the clustering of coreferring event mentions from a news web-crawl. The approach is described in more detail in Section 3.4. Besides the source of unlabeled examples, the data also differs from our augmentation set in size, providing less than one thousand examples for more than half of all ACE event classes.

The only preprocessing of the augmentation and gold-standard corpora we perform is tokenization with the `corenlp` toolkit [Manning et al., 2014].

4.6.2 Evaluation Setup

To evaluate our data augmentation approach we make use of same train/test/dev split[2] of the ACE corpus employed in previous works [Nguyen et al., 2016; Feng et al., 2016; Ferguson et al., 2018]. In addition, we report on the outcome of a 10-fold cross validation experiment, in which the ACE data is split into 10 parts of roughly the same size by randomly selecting entire documents from the original data set. This setup is meant to demonstrate the robustness of the data augmentation approach on data sets with different event distributions.

We report the metrics precision (P), recall (R) and *micro*- and *macro*-averaged F1 scores, presented in Section 3.3.1. As every event type contributes equally to the latter, regardless of the number of examples present in the data set, the macro-averaged F1 score can better demonstrate the effect of the augmentation procedure on less frequent event types.

[2]As this is not a data split provided by ACE, but rather established in related works on event extraction, we provide the exact list of files used in each data set in Table A.1

"Starbucks	is	**taking**	**over**	the	location	in	[..]"	
		Business.Start-Org						
O	O	B-BS	I-BS	O	O	O		$\rightarrow 1tp, 5tn$
O	O	B-BS	O	O	O	O		$\rightarrow 1fn, 1fp, 5tn$

FIGURE 4.3: Event detection evaluation example listing the number of true positives (tp), false positive (fp), and true negatives (tn), for two predictions.

An event mention prediction is considered correct if and only if both the token span and event type of the mention correspond to an ACE event mention. Figure 4.3 provides an example of what constitutes a true positive in two alternative BIO tag sequence predictions.

All results presented in the following chapters are obtained by doing 5 runs of the model with different random sub-samples of augmentation data and random initialization of the model parameters. Different model parameter initializations typically result in fluctuations of the performance of the model. In order to minimize the effect of this factor we report the average f-scores for five runs, together with their corresponding standard deviations σ_{F1}. In addition, we chose to subsample the augmentation data set every time, in order to account for the effect a particular choice of augmentation examples might have on the final score.

Statistical significance testing was performed with a pairwise t-test with Bonferroni correction. Statistically significant ($p<0.05$) improvements in F1 over the baseline are marked (†) in each table.

4.6.3 Implementation Details

Our event tagger was implemented with the `flair`[3] framework, which was designed to facilitate the creation of state-of-the-art sequence labeling, text classification and language models [Akbik et al., 2018, 2019]. Flair builds on top of the PYTORCH library [Paszke et al., 2017], and offers support for a variety of contextualized and non-contextualized word and document embeddings and combinations of them, as

[3] https://github.com/flairNLP/flair/

well as a model training and hyper-parameter selection routines. We make use of the sequence labeling Bi-LSTM model. The framework offers a version of the model with a conditional random field (CRF) loss, which has been successful for other sequence labeling tasks, such as named entity recognition [Lample et al., 2016].

The Bi-LSTM model consists of a single layer with hidden size of 256. We used stochastic gradient descent (SGD) optimization. The best training configuration was determined empirically on the ACE development set, and includes a dropout value of 0.05, learning rate of 0.1, 200 epochs, and mini-batch size of 32. We experimented with the following pre-trained input embeddings: `fasttext` [Bojanowski et al., 2017], `elmo` [Peters et al., 2018] (sizes *original* and *large*[4]), `bert` [Devlin et al., 2019] (sizes *base* and *large*, trained on cased text). We do not perform fine-tuning of the input embeddings when used in conjunction with the Bi-LSTM model.

We compare the performance of the Bi-LSTM event tagger against a `bert` model which was fine-tuned for the task at hand (Section 2.3.3). For this purpose, we adapted the *Token Classification* architecture available in the `huggingface`[5] library. The optimal number of fine-tuning iterations was determined on the development set. We experimented with 3 (recommended for NER [Devlin et al., 2019]) to 20 epochs. The *bert-base-cased* model was used as a starting point. The fine-tuning of *bert-large-cased* failed due to high memory requirements.

4.7 Results

Table 4.2 presents an overview of our best baseline model configurations trained solely on the data available in the ACE corpus.

The first four lines of the table present different configurations of our Bi-LSTM event tagger. The results reveal the advantage of using contextualized input embeddings over the non-contextualized alternative, `fasttext`. The second line of the table includes the best Bi-LSTM model with a CRF output layer. Despite being beneficial for other sequence labelling tasks, the CRF decoder did not prove

[4] *original* refers to the model presented in [Peters et al., 2018], while *large* refers to a model trained on a data set of 5.5B tokens, see https://allennlp.org/elmo.

[5] https://github.com/huggingface/transformers

		Trigger Identification & Classification			
Model		P	R	F1$_{micro}$ (σ_{F1})	F1$_{macro}$ (σ_{F1})
Bi-LSTM	BASELINE$_{fastText}$	63.3	67.5	65.3 (0.4)	60.9 (1.5)
	BASELINE$_{elmo,CRF}$	66.8	**71.0**	68.8 (1.1)	65.3 (2.0)
	BASELINE$_{bert}$	68.7	69.8	69.1 (0.9)	61.8 (2.2)
	BASELINE$_{elmo}$	**69.0**	70.0	**69.5** (0.6)	**67.7** (2.7)
	BASELINE$_{bert,fine\text{-}tune}$	66.3	70.4	68.3 (0.2)	65.4 (2.5)
	CRF [Ferguson et al., 2018]	62.9	70.0	66.3	n/a
	JRNN [Nguyen et al., 2016]	66.0	73.0	69.3	n/a
	HNN [Feng et al., 2016]	84.6	64.9	73.4	n/a

TABLE 4.2: Performance of different configurations of our BASELINE event tagger on the test set of ACE 2005. The best baseline performance per metric is marked in bold. For comparison, we include the scores reported for other state-of-the-art event detection models (Section 3.3.2).

useful for our purposes. The third line presents the best Bi-LSTM model in conjunction with a `bert` input embedding. For comparison, we include a fine-tuned `bert` model (BASELINE$_{bert,fine\text{-}tune}$), which does not utilize a Bi-LSTM architecture.

The best micro- and macro-F1 scores were achieved by the model which employs `elmo` input embeddings (size *original*, see Section 4.6.3). In terms of micro-averaged F1 score, the model is comparable to JRNN [Nguyen et al., 2016], albeit employing a much simpler architecture (Section 3.3.2), and outperforms the CRF model of [Ferguson et al., 2018]. BASELINE$_{elmo}$ is used as our baseline for further comparisons throughout the rest of this chapter.

Table 4.3 presents the evaluation of our data augmentation approaches (Section 4.5) on the test set of the ACE corpus.

Introducing a loss scaling parameter led to better performance of the model over simply concatenating the augmentation and training data (AUG$_{concat}$).

All augmentation strategies led to an improvement in the recall of the model, but affected its precision negatively. This is somewhat to be expected, as we are adding a large amount of new examples for each event type – for half of the event types in ACE we increase the amount of training examples 15 fold or more. At the same time, the process is bound to introduce some errors, as we rely on an

		Trigger Identification & Classification			
Model		P	R	$F1_{micro}$ (σ_{F1})	$F1_{macro}$ (σ_{F1})
Bi-LSTM	BASELINE$_{elmo}$	69.0	70.0	69.5 (0.6)	67.7 (2.7)
	AUG$_{concat}$	65.3	**74.3**	69.5 (0.2)	**70.0** (1.2)
	AUG$_{uniform,\ \lambda=0.1}$	66.9	**73.6**	**70.1** (0.3)	**71.8**† (0.7)
	AUG$_{weight}$	68.8	**71.7**	**70.2** (0.9)	**71.0** (1.2)
	AUG$_{rev.weight}$	66.6	**73.6**	69.9 (0.3)	**70.5** (1.1)
	Ferguson$_{uniform}$	67.7	59.7	63.4 (0.6)	53.2 (1.6)
	trained on aug.data	68.6	66.5	67.5 (0.8)	64.0 (1.5)

TABLE 4.3: Augmentation results on the test set of ACE 2005. Improvements over the baseline metrics are marked on bold. Statistically significant improvements over the baseline are marked with †.

event extraction system trained on few examples to obtain the augmentation data in the first place.

The data augmentation strategy with uniform loss scaling (AUG$_{uniform}$) was most successful out of the augmentation approaches. It achieved a statistically significant improvement over the baseline in terms of macro-averaged F1 score. It improves over the baseline by 4.1 percent with a lower σ_{F1} compared to the other augmentation approaches. This illustrates that our augmentation model works especially well on low-frequency events. A more detailed breakdown is shown in the two rightmost columns of Table 4.5.

The evaluation setup Ferguson$_{uniform}$ presents the performance of our model when the best augmentation strategy (AUG$_{uniform}$) is employed together with the augmentation data set from [Ferguson et al., 2018]. This experiment aims to provide some form of a comparison between the two approaches for the extraction of additional labeled examples, our self-training and their clustering of co-referring event mentions in news articles. Utilizing Ferguson et al. [2018]'s data lead to degradation in performance of the event tagger compared to the baseline, suggesting that their data set introduces a higher number of errors.

The last row of our Bi-LSTM model variants presents the outcome of an experiment in which only augmentation data was used to train our event tagger. We constructed a pseudo training set by randomly including a similar number of mentions per event type as in the original ACE training set (Table 4.1). The model

	Trigger Identification & Classification			
Model	P	R	$F1_{micro}$ (σ_{F1})	$F1_{macro}$ (σ_{F1})
$BASELINE_{elmo}$	71.4	66.8	69.0 (1.6)	58.1 (5.2)
$AUG_{uniform,\ \lambda=0.1}$	70.8	**69.3**	**70.0**† (1.7)	**61.1**† (4.6)

TABLE 4.4: 10-fold cross validation result. Statistically significant improvements over the baseline are marked with †.

performs nearly as good as the baseline trained on the gold data set in terms of micro-averaged F1 score, which suggests that our augmentation data is of very good quality.

We carried out an additional experiment aiming to evaluate our best augmentation strategy on data sets with different event distributions. For this we generated 10 random splits of the ACE corpus, and performed 10 separate experiments in which each split served as a test set once. Furthermore, we used one split as a development set, and the remaining 8 as the training set, so that the sizes of all three sets remained comparable to those of the original ACE split. The complete augmentation data extraction process (Sections 4.4 to 4.5) was repeated for every fold, with an event tagger trained on the training set corresponding to that fold.

Table 4.4 summarizes the outcome of the 10-fold cross-validation experiment. We compare the performance of our best baseline model and augmentation strategy from Table 4.3. The scores achieved for each of the 10 runs of the model have higher variance in terms of macro-averaged F1 score, due to the differences in event distributions in each fold. Despite these differences, the model performance remains robust. Our best results on the ACE data is achieved in this setting, with 1 and 3 percent statistically significant improvement in micro- and macro-averaged F1, respectively.

4.7.1 Error Analysis

Figure 4.4 presents an analysis of the errors made by our baseline and best augmentation model on the ACE test set, focusing on the 8 general ACE event types. In order to enable this comparison we have selected the model predictions from the best run out of the five averaged in Table 4.3 for each model configuration[6].

[6]These correspond to 70.6 micro-F1 and 71.7 macro-F1 for the best $AUG_{uniform}$ model, and 70.2 micro-F1 and 67.5 macro-F1 for the best $BASELINE_{elmo}$ model.

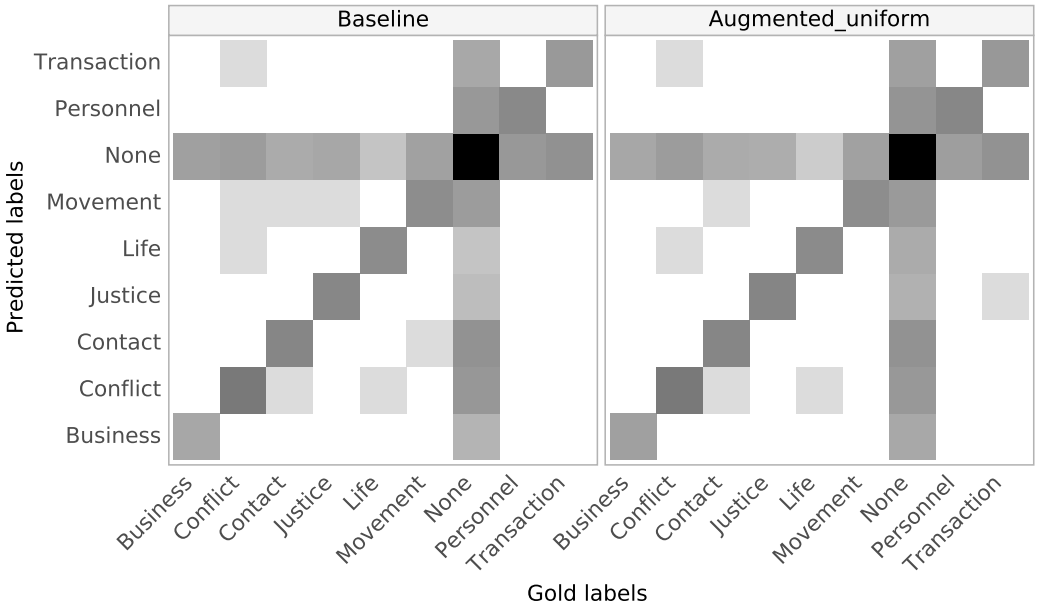

FIGURE 4.4: Error distribution with respect to the general ACE event categories. The density of color denotes the frequency of error, ranging from light gray (fewer errors) to black (a high number of errors). The counts are in the log space in order to account for the disparity of number of instances of each class.

The figure makes evident that the highest number of false positives and false negatives for both models were due to misclassifications of the *None* class (*O*-tag). The augmentation model, however, achieved a reduction of the misclassifications between the other classes (for *Movement*, *Justice*, *Contact*, and *Conflict* events).

In line with the increased recall of the augmentation model, it performed worse in terms of false positives (+19), but led to a reduction of the number of false negatives (−14) compared to the baseline. It outperformed the baseline in terms of true positives and true negatives by 13 decisions each.

The F1 scores achieved by both models on the fine-grained ACE event types are presented in Table 4.5. Surprisingly, the augmentation data was beneficial for the two most frequent ACE event types, *Conflict.Attack* and *Movement.Transport*. As for the infrequent event types, the augmentation approach improved the performance on 6 of the events with less than 50 examples in the original training data (*Business.End-Org*, *Business.Start-Org*, *Justice.Appeal*, *Justice.Extradite*, *Life.Be-Born*, *Personnel.Nominate*). The performance for the remaining 7 remained unchanged, while in only one case we observe a decrease in F1 score (*Life.Divorce*). This goes to show that our augmentation approach is beneficial for supporting the underrepresented event types in ACE.

Event Type	ACE.TRAIN	ACE.TEST	$F1_{BASE}$	$F1_{AUG}$
Business.Declare-Bankruptcy	40	2	100	100
Business.End-Org	31	5	57.1	**76.9**
Business.Merge-Org	14	0	–	–
Business.Start-Org	29	18	40	**43.8**
Conflict.Attack	1273	93	78.5	**81.3**
Conflict.Demonstrate	65	7	66.7	58.8
Contact.Meet	200	50	69.9	69.4
Contact.Phone-Write	112	8	40	36.4
Justice.Acquit	5	1	0	0
Justice.Appeal	30	6	80	**83.3**
Justice.Arrest-Jail	78	6	100	100
Justice.Charge-Indict	96	8	93.3	88.9
Justice.Convict	64	6	90.9	90.9
Justice.Execute	14	2	66.7	66.7
Justice.Extradite	6	1	0	**100**
Justice.Fine	22	6	92.3	92.3
Justice.Pardon	2	0	–	–
Justice.Release-Parole	46	1	100	100
Justice.Sentence	84	11	85.7	**90**
Justice.Sue	60	4	57.1	57.1
Justice.Trial-Hearing	103	5	90.9	83.3
Life.Be-Born	47	3	80	**100**
Life.Die	524	17	85.7	83.3
Life.Divorce	20	9	88.9	84.2
Life.Injure	127	1	100	66.7
Life.Marry	73	10	100	87
Movement.Transport	611	48	64.7	**66**
Personnel.Elect	162	16	81.1	68.2
Personnel.End-Position	159	22	55.3	**57.8**
Personnel.Nominate	11	1	0	**50**
Personnel.Start-Position	92	13	46.2	**53.8**
Transaction.Transfer-Money	128	14	34.8	**40**
Transaction.Transfer-Ownership	92	30	46.2	**47.3**

TABLE 4.5: Event types distribution in ACE 2005 training and test data. The last two columns present a comparison between the performance of the models BASELINE_{elmo} and $\text{AUG}_{uniform}$.

4.8 Conclusion

This chapter presents an approach for the automatic extraction of additional labeled data for the task of event extraction through self-training. A Bi-LSTM event tagger trained on the ACE 2005 corpus is used to perform event trigger identification and classification in a large unlabeled data set. In order to keep the tagger

errors to a minimum, we employ a filtering strategy and retain only those classification decisions which exceed a certain threshold of confidence. The resulting data set is used to enhance the existing gold data in several ways. Our best strategy involves reducing the effect of the augmentation data during re-training through a loss scaling parameter.

Our best augmentation model achieved a statistically significant improvement over the baseline on the ACE test set (macro-F1), as well as in a 10-fold cross validation (micro- and macro-F1) evaluation. Our error analysis revealed that the augmentation approach is especially beneficial for the classification of less frequent event types.

A possible future direction for this work involves the experimentation with different approaches to selecting which of the automatically labeled examples should be assigned to the augmentation data set. In the current set of experiments, the selection is done by randomly sub-sampling instances labeled with high levels of confidence by our event tagger. However, a number of more advanced strategies are possible.

Chen et al. [2018] propose a deep reinforcement learning model which automatically learns an instance selection strategy for self-training. The selection or rejection of an automatically-labeled instance is represented as a decision process which is rewarded based on the performance of a model trained on data including this instance, evaluated on a development set. The decision process takes into account the characteristics of the given instance (its sentence representation), as well as the original model's confidence estimate.

Hedderich and Klakow [2018] present an approach for learning from noisy data (e.g., produced by distant supervision), which can be applied to low-resource scenarios. We face a data sparsity issue as well, as many of the ACE event types are under-represented in the corpus. The proposed architecture aims to learn information about the confusion between clean and noisy labels during training with a mixture of a small amount of clean data and larger amounts of possibly noisy data. This is accomplished via a dedicated noise layer added to the neural network architecture.

Rather than attempting to learn from noisy data, another strategy involves detecting and possibly removing noisy instances in a separate step. For our purposes, this can be done prior to selecting examples for retraining. Wang et al. [2019]

propose to estimate a confidence score for each instance in a training set and train on a re-weighed version of the data, similarly to how we manipulate the training loss when training on the augmentation data set. In order to estimate these confidence scores, they train multiple instances of their model in a cross-validation manner, and employ a voting strategy on the decisions of these models to estimate the likelihood of an example from the test fold being erroneous.

Another possible future direction for this work involves the discovery of event instances with higher variability. A known drawback of self-training is that the automatically labeled instances may be very similar to those already present in the original data set used for training the model [Liao and Grishman, 2011]. This is likely to be the case especially for the model's most confident predictions. Thus, confidence alone may not be the best indicator for selecting examples for the augmentation data set. One way in which more variability can be introduced is to extract the augmentation data from clusters of documents which discuss the same events. For instance, Ferguson et al. [2018] extract clusters of coreferring event mentions from news-wire articles published on the same date which involve the same named entity participants. The resulting clusters likely contain different ways of expressing the same event. High confidence event tagger decisions for some of the mentions in a cluster can prompt the incorporation of the complete set of event mentions from that cluster into the augmentation data set, thus introducing more variability.

Chapter 5

Background and Related Work in Coreference Resolution

5.1 Resolving Reference

When people communicate in spoken or written form, they produce utterances which form a unified whole, rather than a collection of isolated sentences. Each consecutive sentence is semantically linked to its preceding context, and its interpretation often depends on that context. The mechanisms which enable this connectedness are collectively referred to as *cohesion* [Halliday and Hasan, 1976]. Mitkov [2002] notes that cohesion involves the use of abbreviated or alternative linguistic forms which refer to or replace previously mentioned items in the discourse, but can nonetheless be easily recognised and understood by the hearer or reader.

Consider the example sentences in Figure 5.1 which form a cohesive text. Each consecutive sentence relies on its preceding context for interpretation. Sentence (b) cannot be completely understood if considered in isolation from (a). The tokens "it" and "one" serve as placeholders or abbreviations *referring* to something introduced previously. The same applies to sentence (c), whose interpretation is dependent on both (a) and (b) – the reader needs to take into account the previous sentences in order to infer who "they" are. *Reference* is one of the main mechanisms of cohesion identified by Halliday and Hasan [1976], and will be the focus of this chapter.

(a) John likes the red car.

(b) Mary also likes *it*, but prefers the green *one*.

(c) While *it* was not *their* first choice, the yellow hatchback was all *they* could afford.

FIGURE 5.1: Example of cohesive text.

5.1.1 Anaphora

The sample text in Figure 5.1 contains a form of reference known as *anaphora*, which can be defined as a "cohesion which points back" [Halliday and Hasan, 1976]. The expression which points to a previously mentioned item is called an *anaphor*, while the expression which the anaphor points back to is known as an *antecedent*. For instance, the antecedent of the anaphor "it" in (b) is "the red car" in (a), while the anaphor "one" has as an antecedent the token "car".

John likes the red car. Mary also likes *it*, but prefers the blue *one*.

Pronoun anaphors are part of the more general category of nominal anaphora, along with proper names, nominal descriptions, and demonstratives. Nominal anaphora is the most abundant and widely studied type of reference [Mitkov, 2002]. Verbs and adverbs are among the other common grammatical categories of anaphors.

Zero anaphora occurs when an anaphor is not explicitly present in the text, but is nonetheless understood by the reader and relies on an antecedent expression for recovering its meaning. Mitkov [2002] notes that this type of anaphor enhances the coherence of a sentence, and is the most concise or abbreviated linguistic form anaphors can take. In (b), it is understood that Mary is the one who prefers the blue car.

Mary also likes it, but ∅ prefers the blue one.

When an expression points to something mentioned later in the discourse as opposed to previously, it is known as a *cataphor*. An example for this is present

in sentence (c), where the pronoun "it" occurs before its antecedent "the yellow hatchback".

> While *it* was not their first choice, the yellow hatchback was all they could afford.

In some cases the antecedent of an anaphor may consist of several, non-consecutive expressions, known as *split antecedent*. For instance, the anaphoric pronouns "their" and "they" in (c) refer to the group consisting of John and Mary.

> John likes the red car. Mary also likes it, but prefers the blue one. While it was not *their* first choice, the yellow hatchback was all *they* could afford.

Anaphora resolution in NLP is the process of automatically determining the antecedents of each anaphor (and cataphor) in a text. *Anaphoricity detection* is sometimes performed as an additional step prior to anaphora resolution. The aim of this task is to identify and filter non-anaphoric noun phrases, such as the non-anaphoric uses of the pronoun *it* – pleonastic it ("*It* is raining"), clefts ("*It* was John who loved Mary"), etc., as well as some generic, underspecified, and abstract expressions.

5.1.2 Coreference

When the anaphor and its antecedent refer to the same object, person, organisation, or other real-world entity, they are said to *corefer*. The expression which is utilised to refer to this entity is called a *referring expression*, while the entity itself is known as the *referent*. A set of expressions with a common referent forms a *coreference chain*.

The referent is not to be confused with the antecedent. The antecedent is a previous textual mention, upon which the anaphor relies for its interpretation, while the referent is the actual entity in the world which is being discussed. The antecedent of "it" (b) is "the red car" (a), while the referent of both is the actual red car entity in the world. Thus, this anaphor and its antecedent are also referring expressions, and are said to exhibit the identity of reference relation [Mitkov, 2002].

However, not every anaphor and antecedent pair are coreferential. Certain types of anaphors denote an entity of a similar description, rather than one of the same identity. This is known as *identity of sense* anaphora [Mitkov, 2002]. The anaphor "one" (b), for instance, is not coreferential with its antecedent "car" (a) – the red car and the blue car denote two different entities in the world. Just as not every anaphor is coreferential, not every coreferential expression is anaphoric. Two mentions which appear in separate documents can still be coreferential, when they have the same referent[1]. Anaphora, on the other hand, is always limited to the current discourse context.

Coreference resolution is an NLP task aimed at the automatic detection and grouping of textual mentions of real world entities. The task is sometimes referred to as noun phrase coreference resolution, as it primarily focuses on nominal referring expressions. Verbal coreference occurs when two or more expressions refer to the same event. Verbal coreference resolution is more frequently denoted as *event coreference*, or *event linking* (see Chapter 3).

The coreference resolution output corresponding to the example in Figure 5.1 would include the following three[2] coreference chains: {"*the red car*", "*it*"}, {"*it*", "*the yellow hatchback*"}, and {"*their*", "*they*"}.

Anaphora and coreference resolution are two closely related tasks – despite the aforementioned differences between the two phenomena, the large majority of referring expressions are also anaphoric. Coreference resolution systems have drawn inspiration from classical works in anaphora resolution. This is visible in rule-based approaches to the task [Lee et al., 2011], as well as the choice of features for supervised machine learning models (e.g., agreement, syntactic constraints, etc.). Anaphoricity detection has also been successfully applied to coreference resolution [Ng and Cardie, 2002a; Ng, 2004]. A distinction between the two tasks is that anaphora resolution is typically performed in terms of pairwise decisions on candidate ⟨anaphor – antecedent⟩ pairs. Coreference resolution, on the other hand, involves assigning a much larger set of entities into coreference chains, resulting in an explosion of possible combinations. In addition, the repeated reference to the same entity can often result in long-distance dependencies, further complicating the task (see Section 5.1.3).

[1]Identifying such expressions falls under the task of *cross-document* coreference resolution.

[2]Due to its complexity, zero anaphora resolution is usually considered as a separate task and beyond the scope of coreference resolution.

Another task related to coreference resolution is *entity linking*. While the former focuses on identifying and grouping all textual mentions of the same entity, the latter goes a step further in that it aims to identify the referent of those entity mentions. Determining the identity of the referent constitutes in tagging entity mentions with unique identifiers which link them to a knowledge base. For instance, the textual mention *Madonna* could be linked to the Wikipedia page dedicated to the singer, or to the one describing the biblical figure, in order to identify its referent. Entity linking primarily focuses on named entities, and does not aim to identify *all* mentions of these entities in the text.

5.1.3 Referring Expressions

Spoken and written communication usually involves the repeated reference to the same discourse entities. While equivalent in terms of their referent, the referring expressions employed in this process are typically not equivalent amongst each other in terms of generality and amount of information about the referent which they encode.

Consider the text snippet in Figure 5.2, where all the mentions referring to a company are marked. The resulting coreference chain contains a wide range of expressions – proper noun phrases, definite noun phrases, and pronouns. The pronoun *it* provides little information about the referenced entity, namely that it is *neuter* and *singular*. The noun phrase *the company*, on the other hand, tells the reader about the category of the entity being discussed, in addition to its number and gender. Full forms of proper names, such as *Elco Industries Inc.*, are considered among the most specific forms referring expressions can take [Ariel, 1990; Almor, 1999]. Even though some referential expressions, such as pronouns, are very general and are thus applicable to a wide range of entities, the reader is typically able to pick out the correct referent without difficulty. The repeated use of the most specific referring expression form available to point to an entity, such as a proper name, would virtually eliminate any ambiguity in its resolution. Nonetheless, such expressions seem to be reserved for specific places in the discourse. The factors which guide the choice of referential form have been of interest not only from a theoretical perspective, but also for the design of computational models for coreference and anaphora resolution.

Elco Industries Inc. said *it* expects net income in the year ending June 30, 1990, to fall below a recent analyst's estimate of $1.65 a share. *The Rockford, Ill., maker of fasteners* also said *it* expects to post sales in the current fiscal year that are "slightly above" fiscal 1989 sales of $155 million. *The company* said *its* industrial unit continues to face margin pressures and lower demand. In fiscal 1989, *Elco* earned $7.8 million, or $1.65 a share.

FIGURE 5.2: Example of different types of co-referring mentions of an entity from the OntoNotes corpus [Hovy et al., 2006].

Discourse *salience* is among the main factors which are considered to influence the choice of referential form [Almor, 1999]. Entities which are more salient or prominent in the discourse tend to be referenced by shorter, more general forms such as pronouns, while definite descriptions and proper names are usually reserved for reintroducing less salient entities. Almor [1999] traces this to the cooperative principles for effective communication [Grice, 1975] – a referring expression should be as informative as necessary in the given context in order to enable the successful identification of the referent by the reader, but not more informative than necessary. An exception to this can occur if the expression serves the additional purpose of introducing new information about the referent. For instance, *The Rockford, Ill., maker of fasteners* (Figure 5.2) tells us about the location and product offered by the company, while being utilized to refer to a very salient entity. This contrasts with the antecedent–anaphor relationship discussed previously, which suggests that the anaphor is a more abbreviated form compared to its antecedent.

Ariel [1990] puts forward the view that referential expressions serve the function of accessibility markers, instructing the reader or listener on how much effort is needed to retrieve from memory who the referent is. The less accessible the referent, the more elaborate the referential marker would be used to point to it, and vice versa. A number of factors are considered to influence the degree of accessibility of an entity in addition to its *salience*. The larger the *distance* separating different mentions of the same entity, the lower the degree of accessibility if its mental representation, which could be attributed to the decay of information in working memory [Almor, 1999]. If the two expressions are located within the same textual or discourse unit (e.g., a paragraph), they are considered to exhibit a

higher degree of *cohesion* and hence accessibility. Finally, *competition* refers to the number of competitors on the role of antecedent – the more potential antecedents there are, the lower the degree of accessibility each would have.

It becomes evident that the distribution of referential forms is not arbitrary, but follows specific patterns which could be attributed to a multitude of factors. A general tendency seems to be that in cases when the identity of the referent cannot be established unambiguously (e.g., due to competing referents, distance, etc.), the speaker or writer might choose a more specific referential form in order to reintroduce the referent into the discourse. In contrast, when the referent is easy to infer, the chosen expression can be abbreviated to a pronoun, or even a zero anaphor, following the principles of effective communication. Beyond a means for identifying the referent, however, a referring expression can be utilised to enrich the mental representation of the referenced entity by introducing additional information about it. Such cases pose exceptions to the aforementioned generalisation about referential form usage.

5.2 Coreference Corpora

The OntoNotes corpus [Hovy et al., 2006; Weischedel et al., 2011] is the largest and most widely used coreference data set for English. It was central to the CoNLL 2011 and 2012 [Pradhan et al., 2011, 2012] shared tasks on modelling unrestricted coreference, which established an evaluation protocol serving as the standard for comparing coreference resolution models to this day. The English portion of the OntoNotes corpus comprises roughly 1.5 million words of news-wire, magazine articles, broadcast news, broadcast conversations, web data, conversational speech data, and biblical text.

According to the annotation guidelines of OntoNotes, coreference involves linking "all the specific mentions in a text that point (refer) to the same entities and events". The following lines summarize some of the specificities of the coreference annotations of the corpus.

Entity mentions usually take the form of pronouns, noun phrases, or proper nouns. All personal and possessive pronouns and demonstratives are linked to their antecedents with the exception of expletive or pleonastic pronouns and the generic *you*. Noun phrase mentions typically consist of, but are not limited to, definite

	TRAIN		DEV		TEST	
category	count	%	count	%	count	%
noun phrase	61.8K	39.46	9.7K	45.57	9.2K	42.97
pronoun	66.7K	42.61	7.8K	36.66	8.2K	38.69
proper noun	18.1K	11.60	2.2K	10.66	2.3K	10.96
other noun	2.636	1.68	546	2.55	500	2.33
verb	2.522	1.61	299	1.40	342	1.60
other	4.761	3.04	676	3.16	738	3.45

TABLE 5.1: Distribution of mentions in the training, development, and test splits of the English portion of OntoNotes by their syntactic category, as presented in [Pradhan et al., 2012].

descriptions. In complex nested noun phrases such as the example below, the longest logical span participates in coreference with other mentions of the same entity:

> There's already word of ⟨⟨⟨a possible Israeli-Palestinian **summit**⟩ in Egypt⟩ in the next several days⟩$_1$. ⟨The summit⟩$_1$ will [..]

Generic, abstract, or underspecified nominal mentions can be linked with referring pronouns and other definite mentions, but not with other generic nominal mentions:

> ⟨Meetings⟩$_1$ are most productive when ⟨they⟩$_1$ are held in the morning. However, ⟨meetings⟩$_2$ can have low attendance if ⟨they⟩$_2$ are held too early.

Coreference between event mentions is also marked. The heads of verb phrases can be coreferent with a noun phrase in cases of nominalizations (e.g., *grow* and *growth*), or with another verb. Event mentions are marked only if they participate in event coreference. Unlike the ACE 2005 corpus (Section 3.2), event arguments and event types are not annotated. Table 5.1 summarizes the distribution of mentions in the corpus with respect to their syntactic category.

OntoNotes makes a distinction between identity coreference and appositive reference. Appositive constructions involve a different form of reference – between

the head of the appositive and the remaining noun phrases in the construction, which function as attributes of the head. Appositive attributes cannot participate in identity coreference on their own and are not coreferent with the head. The longest span rule applies here as well when linking appositives with other coreferent entity mentions:

> $\langle\langle$John$\rangle_{head}$, $\langle$a linguist I know$\rangle_{attribute},\rangle_1$ is coming to dinner. $\langle$He$\rangle_1$ will [..]

Copular constructions and small clauses are also considered to introduce attributes rather than mentions of the same entity. For instance, the predicate of the copula, "a linguist I know", does not participate in a coreference chain with the subject "John" in the following example:

> $\langle$John$\rangle_1$ is *a linguist I know.* $\langle$He$\rangle_1$ is coming to dinner.

Other coreference resolution corpora include the Message Understanding Conference (MUC) data sets, MUC6 [Chinchor and Sundheim, 2003] and MUC7 [Chinchor, 2001], and the ACE (2000-2005) data sets. The MUC corpora contain annotations for unrestricted noun phrase coreference, but have the disadvantage of being relatively small in size – roughly 25K words for MUC6, and 40k for MUC7 [Pradhan et al., 2012].

The ACE corpora were used in the *Entity Detection and Tracking* tasks of the ACE Program. While the corpora is larger in size, the annotations are limited to coreference for a set of seven entity types: person, organization, location, weapon, vehicle, facility, and geopolitical entity. Unlike MUC and OntoNotes, the ACE corpus contains annotations for *singletons*, or entities mentioned only once.

In addition to their limitations in terms of size and coverage of entity types, the annotation decisions made in the two data sets have been criticised as being inconsistent with linguistic theory of coreference. The definition of coreference adopted in MUC has been deemed unclear, mixing elements of anaphora and predication [van Deemter and Kibble, 2000]. Nominal predicates in copula constructions are treated as coreferring with the subject in both MUC and ACE. Similarly, the attributes of appositives are deemed coreferent with the head of the construction. Thus, both corpora would consider "John" and "a linguist I know" in the two previous examples to be separate mentions of the same entity.

5.3 Coreference Evaluation

A number of metrics have been proposed for the evaluation of coreference resolution. The list includes, among others, MUC [Vilain et al., 1995], B^3 [Bagga and Baldwin, 1998], $CEAF_m$, $CEAF_e$ [Luo, 2005], BLANC [Recasens and Hovy, 2011], and LEA [Moosavi and Strube, 2016].

Despite the variety of available metrics, there is not a single widely agreed-upon measure employed in the community. Coreference evaluation has proven to be a non-trivial problem. A suitable metric should be intuitive and interpretable, discriminative, and capable of penalising mismatches due to erroneous mention identification in addition to erroneous coreference decisions. The majority of the proposed metrics have been shown to lack in one or more of these desired properties [Stoyanov et al., 2009; Holen, 2013; Moosavi and Strube, 2016].

Due to the lack of a single accepted measure, the CoNLL 2011 and 2012 shared tasks [Pradhan et al., 2011, 2012] have ranked participating systems with an average of the F1 scores produced by three metrics, MUC, B^3, and $CEAF_e$. Moosavi and Strube [2016] rightfully note that taking the average of three unreliable metrics does not produce a reliable one. Furthermore, the use of an average score makes it difficult to compare competing models in terms of precision and recall. Nonetheless, the CoNLL F1 score has since become the standard in coreference evaluation, while the provided scorer implementation [Pradhan et al., 2014] has helped to establish a much needed basis for comparison across existing coreference models. The following lines summarize the component metrics of the CoNLL F1 score, which was also used for the experiments described in Chapter 6.

MUC [Vilain et al., 1995] is one of the earliest metrics proposed for coreference resolution evaluation. It represents coreference decisions in terms of links between coreferring mentions. Precision and recall are estimated as the number of links which need to be added or removed from a system prediction, or the *response*, in order to obtain the gold-standard set of coreference chains, or the *key*. Let $K = \{k_1, .., k_n\}$ denote the set of key coreference chains, and $R = \{r_1, .., r_n\}$ be the set of response chains. MUC recall is computed via Equation (5.1)[3], where $p(k_i)$ are the partitions obtained by intersecting k_i with R. Precision is computed in the same way, but by switching the role of the response and key.

[3]We use the notation of [Moosavi and Strube, 2016] for all metrics.

$$\text{Recall}_{\text{MUC}} = \frac{\sum_{k_i \in K}(|k_i| - |p(k_i)|)}{\sum_{k_i \in K}(|k_i| - 1)} \tag{5.1}$$

MUC has been criticised for its poor discriminative abilities, as it ranks all mistakes equally regardless of the impact they might have on the overall performance [Bagga and Baldwin, 1998]. For instance, it would not distinguish between an erroneous link which merges two singletons, and one which merges the two most prominent entities of the text, which would be much more problematic [Moosavi and Strube, 2016]. Furthermore, being link-based, MUC cannot handle singletons and is thus not suitable for some corpora.

B^3 or BCUB [Bagga and Baldwin, 1998] was proposed as a solution to the aforementioned shortcomings of MUC. It is a mention-based metric which assigns a score to each mention in the key equal to the number of correct mentions in the response entity containing that mention, over the size of the key entity to which the mention belongs. The final recall score (Equation (5.2)) is the normalized sum of the scores for each mention. Precision is computed by reversing the role of response and key.

$$\text{Recall}_{\text{B}^3} = \frac{\sum_{k_i \in K} \sum_{r_j \in R} \frac{|k_i \cap r_j|^2}{|k_i|}}{\sum_{k_i \in K} |k_i|} \tag{5.2}$$

B^3 can produce unintuitive results in some borderline cases [Luo, 2005]. Moosavi and Strube [2016] suggest that the issue stems from the fact that coreference is evaluated in terms of the presence or absence of key mentions in response entities. As a result, a key mention which exists in any of the response entities would receive some credit, even when it is not truly coreferent with any of the remaining mentions in that entity.

The CEAF metric [Luo, 2005] first finds the best one-to-one alignment between key and response entities according to a similarity measure. The optimal alignment (the one which maximizes the total entity similarity) is obtained through the use of the Kuhn-Munkres algorithm. There are two variants of the CEAF metric depending on the similarity measure employed in the process. CEAF_m computes the similarity as the number of common mentions in two entities. The similarity measure employed in CEAF_e is shown in Equation (5.3). Recall is computed

via Equation (5.4), where K^* denotes the set of key entities participating in the optimal one-to-one mapping g^*.

$$\phi(k_i, r_j) = \frac{2 \times |k_i \cap r_j|}{|k_i| + |r_i|} \tag{5.3}$$

$$Recall_{\text{CEAF}_e} = \frac{\sum_{k_i \in K^*} \phi(k_i, g^*(k_i))}{\sum_{k_i \in K} \phi(k_i, k_i)} \tag{5.4}$$

One of the main criticisms of CEAF is that is does not take into account any correct coreference decisions present in the unaligned response entities [Moosavi and Strube, 2016]. Furthermore, all entities are weighed equally, regardless of their impact on the overall performance as determined by their size [Stoyanov et al., 2009].

The CoNLL F1 score [Pradhan et al., 2014] is obtained by averaging the F1 scores of the MUC, B^3, and $CEAF_e$ metrics (Equation (5.5)).

$$\text{F1}_{\text{CONLL}} = \frac{1}{3} \times (\text{F1}_{\text{MUC}} + \text{F1}_{\text{B}^3} + \text{F1}_{\text{CEAF}_e}) \tag{5.5}$$

The BLANC measure [Recasens and Hovy, 2011] is a variation of the Rand index for clustering evaluation applied to the coreference task. BLANC is the only measure of the ones discussed so far which directly rewards correct coreference as well as non-coreference decisions. Each mention is considered as linked to every other mention in the data by either a "coreferent" or "non-coreferent" link. Separate Precision, Recall and F1 values are computed for the two types of links when contrasting a system's predictions against the gold standard. The final BLANC score is the average of the F1 measures for the two.

5.4 Supervised Coreference Resolution

5.4.1 Model Architectures

One of the most widely used coreference resolution architectures is the *mention-pair* model [McCarthy and Lehnert, 1995; Aone and William, 1995; Soon et al.,

2001]. It breaks down the task of coreference resolution into binary classification decisions on pairs of mentions. Subsequently, a clustering algorithm is employed to build coreference chains from those predictions.

The need for clustering stems from the fact the model's predictions alone might not be sufficient to form coreference chains. For instance, it might predict that mention A is coreferent with mention B, and that mention B is coreferent with mention C, but fail to classify A and C as coreferent [Ng, 2010]. In such cases, clustering is needed to enforce the transitivity property of coreference. The two most popular clustering approaches are *closest-first* clustering [Soon et al., 2001] and *best-first* clustering [Ng and Cardie, 2002b]. In the former, a mention is grouped with its closest predicted antecedent, while the latter selects the antecedent deemed most probable by the model.

Another challenge for the mention pair model is the creation of training instances [Ng, 2010]. As the majority of mentions do not stand in a coreference relation, simply pairing any two would lead to a highly skewed class distribution with mostly negative examples. One popular strategy to cope with this issue is to create a positive training instance by pairing a mention m_i and its closest antecedent m_k, and a negative one by pairing mention m_i with any intervening non-coreferent mention m_j $(k < j < i)$ [Soon et al., 2001].

A commonly cited weakness of the mention-pair model is that decisions on pairs of mentions are taken in isolation and cannot benefit from previous predictions or influence future ones. Furthermore, the lack of sufficient evidence to make an informed classification decision for certain pairs can lead to errors.

The *entity-mention* model [Yang et al., 2004; Luo et al., 2004] has been proposed as a way to improve on the expressiveness limitations of the mention-pair model. For each mention, the model decides on whether it should be assigned to an existing partially-formed mention cluster, or start a new cluster. Working with a mention cluster allows for the model to base its classification decisions on information from multiple mentions as opposed to a single candidate antecedent. Cluster-level features can be defined in order to make sure that all mentions in cluster comply to specific constraints, such as agreement in number and gender. Despite the improved expressiveness, the entity-mention model has not lead to significantly improved performance in comparison with the simpler mention-pair model [Ng, 2010].

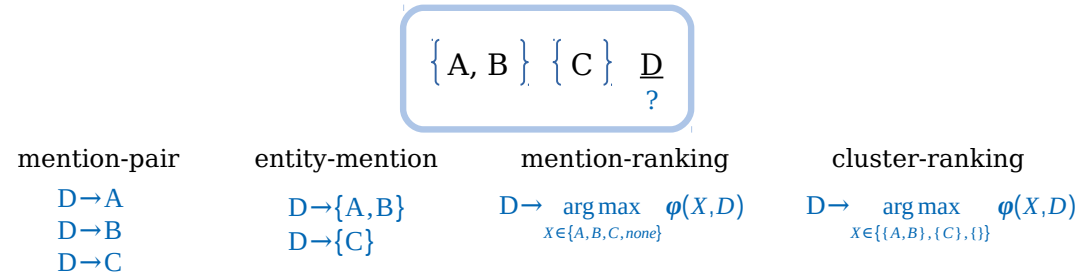

FIGURE 5.3: An example of the decisions each coreference model would take to determine the antecedent of mention D among the candidates A, B, C, or none.

The *mention-ranking* model [Denis and Baldridge, 2007, 2008] offers an alternative approach to coreference resolution. Rather than viewing the task as a classification problem, it computes a ranking of all possible candidate antecedents with respect to the current mention. The mention is then linked to the most probable antecedent. The ranking model has the advantage that it directly captures the competition among potential antecedents, rather than considering them in isolation. This model architecture has been shown to consistently outperform the mention-pair model [Ng, 2010]. Similarly to the way in which the entity-mention model improves on the expressiveness of the mention-pair model, the *cluster-ranking* model proposed by Rahman and Ng [2009] extends on the mention-raking model by enabling the use of cluster-level features and the competition among partially formed clusters with respect to the mention under consideration.

Figure 5.3 presents a comparison between the coreference architectures discussed previously. The figure exemplifies how each model would predict the antecedent of mention D among the candidate antecedents A, B, C, or none (D is the first mention of a new entity). The mention-pair and entity-ranking models perform several binary classifications, while the mention-ranking and cluster-ranking models compute a ranking ϕ over the candidate antecedents, subsequently selecting the highest scoring option among them.

5.4.2 Learning Algorithms

A variety of machine learning algorithm have been employed for the task of supervised coreference resolution. Popular methods include: decision trees [McCarthy and Lehnert, 1995; Soon et al., 2001; Ng and Cardie, 2002b; Yang et al., 2004],

conditional random fields [McCallum and Wellner, 2005], maximum entropy classifiers [Luo et al., 2004; Culotta et al., 2007; Denis and Baldridge, 2008; Björkelund and Farkas, 2012], support vector machines [Finley and Joachims, 2005; Uryupina, 2010], averaged perceptron [Bengtson and Roth, 2008], structured conditional random fields [Durrett and Klein, 2014], and structured perceptron [Björkelund and Kuhn, 2014; Martschat et al., 2015], among others.

More recently, deep learning-based approaches to coreference resolution have gained popularity [Lee et al., 2017, 2018; Kantor and Globerson, 2019; Peters et al., 2018; Fei et al., 2019; Joshi et al., 2019].

For a more detailed overview of supervised coreference resolution learning algorithms we refer the reader to [Ng, 2010], [Zheng et al., 2011] and [Stylianou and Vlahavas, 2019].

5.5 Features for Supervised Coreference Resolution

A coreference resolver typically considers each noun phrase in a given text as a prospective referring expression candidate, and each preceding noun phrase or sets of noun phrases as possible antecedents to that expression. A number of features which aim to determine the compatibility between candidate co-referring mentions have been proposed for the task [Bengtson and Roth, 2008; Recasens and Hovy, 2009; Ng, 2010; Zheng et al., 2011; Durrett and Klein, 2013]. These include unary features modelling the properties of individual mentions, and pairwise features comparing against a single candidate antecedent (mention-pair, mention-ranking model), or against the mentions in partially-formed coreference chains (entity-mention, cluster ranking model). In the latter scenario, the features compute the compatibility with *all*, *most*, or *any* of the members of the chain under consideration.

Section 5.5.1 offers an overview of the most commonly used features in supervised coreference resolution. Section 5.5.2 focuses on the use of semantic knowledge for the task.

5.5.1 Surface Features

Mentions of the same entity typically agree in *number*, *gender*, and *person* . The gender of person names can be determined by a lookup in dedicated name lists [Bergsma and Lin, 2006], or through other heuristics (e.g., title matching), but can prove challenging for common noun phrases and out-of-vocabulary or ambiguous names (e.g., "*Clinton* won the debate"). In addition, special care needs to be taken in the presence of indirect speech, which can introduce exceptions to the agreement rule. In the example below, the correct grouping is between "*she*" and "*I*", regardless of the mismatch in person, and not between "*she*" and ‘‘her", despite the agreement in both person and gender.

"⟨I⟩$_1$ gave ⟨the book⟩$_2$ to ⟨her⟩$_3$ yesterday.", ⟨she⟩$_1$ said.

Some *distance*-based features have been proposed as well. Such features indicate the distance in number of tokens, sentences, or even number of intervening noun phrases (other potential antecedents) between a referring expression and its candidate antecedent. Empirical evidence suggests that the distance between anaphors and their antecedents varies depending on the type of expressions involved [Mitkov, 2002]. Pronouns, for instance, tend to be situated close to their antecedents, and can usually be resolved within the same or the preceding sentence. Proper names and definite descriptions, on the other hand, are usually further away from their antecedents. In some extreme cases, the distance has been reported to reach 30 sentences or more [Ariel, 1990; Mitkov, 2002].

A multitude of *string matching* features have been shown to be particularly effective for named entity and noun phrase mentions [Ng, 2010]. Such features determine if two mentions are an exact or partial string match, which would be helpful in establishing a link between "*Elco*" and "*Elco Industries Inc.*" in the example in Figure 5.2. Beyond surface string match, other related features include *edit distance* and *acronym* matching.

Salience seems to play a major role in the choice of referring expression (Section 5.1.3). Certain types of referring expressions (e.g., pronouns) are more likely to point to very salient entities and would rarely be used to refer to non-salient ones, while others (e.g., names, definite descriptions) are typically used to introduce new entities, or re-introduce less-salient entities. Attempt have been made

towards modelling the salience of entities to serve as feature for supervised coreference resolution. Antecedents which appear in a prominent grammatical role (e.g., in the subject position) or have been mentioned recently, are considered more salient [Ng, 2010]. Repetition can be used as an indicator that an entity is the topic of the document. Iida et al. [2009] employ a dedicated salience ranker model.

Somewhat related are features indicating the type of the referring expressions, with values typically including proper name, definite/indefinite noun phrase, demonstrative, possessive, or other. Besides its usefulness in conjunction with other features, this value can pertain to the typical order of referential expressions in a coreference chain. For instance, a pronoun is not likely to be the first mention of an entity.

5.5.2 Semantic Features

Surface-level features such as the ones described so far do not always provide sufficient information for the successful disambiguation between antecedent candidates. In practice, a text often describes the interactions between multiple referents with similar characteristics (e.g., multiple inanimate objects, persons, etc.). In order to infer the identity of the referents being discussed, the reader relies on their understanding of the semantics of language and their knowledge of the state of the world.

Co-referring mentions may not share any surface string similarity but rather exhibit some form of semantic relatedness. Lexical semantic relations such as *synonymy* and *hyponymy/hypernymy*, are commonly used to establish a link to previous mentions of the same entity. For instance, consider the co-referring expressions in the following passage:

> ⟨**Laika**⟩$_1$ became one of the first animals in space. ⟨The stray **mongrel** from the streets of Moscow⟩$_1$ was selected to be the occupant of the Soviet spacecraft Sputnik 2. ⟨The **dog**⟩$_1$ was launched into outer space on 3 November 1957.

The dog breed "*mongrel*" constitutes a (class) hyponym of "*dog*", while "*Laika*" is an instance hyponym[4] of both. Another example is present in Figure 5.2: "*Inc.*" (incorporated) and "*maker of fasteners*" can both be considered hyponyms of "*company*", while "*Elco*" and "*Elco Indusitries Inc.*" are synonyms and instances of the latter.

In certain cases it may not be sufficient to consider the candidate referring expressions in isolation. Instead, a broader sentential context may need to be taken into account to make the correct prediction. Consider the example below, which offers two continuations of the sentence "*My dog Sparky hates the mailman*":

(1) ⟨My dog Sparky⟩$_1$ hates ⟨the mailman⟩$_2$. ⟨He⟩$_1$ tried to *bite* ⟨him⟩$_2$ on multiple occasions.

(2) ⟨My dog Sparky⟩$_1$ hates ⟨the mailman⟩$_2$. ⟨He⟩$_2$ tried to *kick* ⟨him⟩$_1$ on multiple occasions.

Sparky and the mailman are both plausible antecedent candidates for "*He*" and "*him*" in each consecutive sentence, as they exhibit no surface disagreement and match in number, person and gender with the two referents. The key to the successful disambiguation of the referents lies in determining the *selectional preferences* of the verb predicates in the second sentence. As dogs are more likely to bite humans than the other way around, we can deduce with some certainty who tried to bite whom in (1). Analogously we can determine who tried to kick whom (2), as a dog's preferred form of attack typically does not involve kicking.

Consider a different example from [Ponzetto and Strube, 2006] which further illustrates the importance of contextual semantic information:

> A state commission of inquiry into the sinking of the Kursk will convene in Moscow on Wednesday, ⟨the Interfax news agency⟩$_1$ *reported*. ⟨It⟩$_1$ *said* that the diving operation will be completed by the end of next week.

[4]Class hyponymy refers to the relationship between common nouns, which can be seen as labels of classes (e.g., *folk singer* is a hyponym of *singer*). Instance hyponymy describes the relationship between a class and a specific entity (or instance) of that class (e.g., *Johny Cash* is an instance of *singer*) [Miller and Hristea, 2006].

There are multiple candidate antecedents for the pronoun "*It*": the comission, the sinking, the Kursk, Moscow, the news agency. Agreement and surface features fail to provide sufficient discriminative information for its disambiguation. Furthermore, both the commission and Moscow[5] could be considered good candidates according to the selectional preferences of its predicate "said". The correct link to the antecedent "*the Interfax news agency*" can be established by inferring that the two expressions are agents of the semantically-related predicates *report* and *say* [Ponzetto and Strube, 2006].

In certain cases the cues for the resolution of referring expressions are extra-linguistic. When selecting a referring expression, the reader or speaker may rely on common knowledge about prominent entities, such as the fact that the singer Madonna is also known as the "Queen of pop", or that "Emerald Isle" is the poetic name for Ireland. Consider the example from [Mitkov, 2002]:

> ⟨Tony Blair⟩$_1$ met ⟨President Yeltsin⟩$_2$. ⟨The old man⟩$_2$ had just recovered from a heart attack.

The correct resolution of "*The old man*" is dependent on the information that one of the two candidate referents is older compared to the other. It is unlikely that this information has been explicitly noted in the text, however, as it was common knowledge for the intended audience at the time of writing.

A long line of work has been devoted to the incorporation of semantic and world knowledge into coreference resolution. Some of the investigated sources of such information include lexical databases (WordNet [Miller, 1995], FrameNet [Baker et al., 1998], VerbNet [Schuler and Palmer, 2005], PropBank [Palmer et al., 2005]), structured and semi-structured encyclopedic resources (Wikipedia, Yago [Suchanek et al., 2007], DBPedia [Auer et al., 2007], FreeBase [Bollacker et al., 2008]), and semantic knowledge extracted automatically from unstructured data.

WordNet is a popular source of lexical semantic information for the task, in particular for the resolution of common nouns [Vieira and Poesio, 2000; Soon et al., 2001; Harabagiu et al., 2001; Ng and Cardie, 2002b; Markert and Nissim, 2005; Daumé III and Marcu, 2005; Ponzetto and Strube, 2006; Bengtson and Roth, 2008; Zhiheng et al., 2009]. The WordNet taxonomy is used to assign coarse-grained semantic classes to common noun referring expressions (e.g., "person", "object", etc.),

[5] when used as a metonym to refer to the government of the country.

which can be used alongside the class assigned to proper nouns by a named entity tagger. The WordNet semantic class feature is obtained by traversing the taxonomy tree via the hypernymy relation. An alternative use case is to directly determine if two expressions are related via synonymy or hypernymy [Vieira and Poesio, 2000; Bengtson and Roth, 2008]. The lexical data base can also be used to compute similarity measures based on the distance between two nouns in the taxonomy tree [Daumé III and Marcu, 2005; Ponzetto and Strube, 2006].

A challenge in the application of WordNet to the task is the need to determine the correct synset a token belongs to prior to querying the taxonomy tree. The word sense disambiguation step is frequently circumvented by only considering the first (i.e., most frequent) sense of a token [Soon et al., 2001; Ng and Cardie, 2002b], or by using information from all of the synsets [Ponzetto and Strube, 2006]. For instance, Bengtson and Roth [2008] check whether any sense of a mention's head noun phrase is a synonym, antonym, or hypernym of any sense of the other mention's head.

Another drawback of using WordNet for coreference resolution is its limited coverage of named entities and specialized concepts [Strube and Ponzetto, 2006; Versley Y., 2016]. WordNet distinguishes between *class* and *instance* hyponymy [Miller and Hristea, 2006]. Both relationships are of interest to coreference resolution, however only the former is well represented in the data base. For instance, only 48 instances of *country* and 32 instances of *singer* are included. The lack of good coverage of named entities and specialized terms has prompted the exploration of encyclopedic resources for the task.

Wikipedia offers a very large collection of encyclopedic knowledge from various domains. Due to its collaborative nature, its content is kept up to date and continuously extended to incorporate emerging entities. For these reasons, Wikipedia constitutes an attractive source of knowledge for coreference resolution [Strube and Ponzetto, 2006; Ponzetto and Strube, 2006; Bryl et al., 2010b; Uryupina et al., 2011; Ratinov and Roth, 2012]. Similarly to WordNet, Wikipedia provides a taxonomy-like structure by means of its category tree, a hierarchy of the categories assigned to articles. It also offers structured information in the form of info-boxes, tables, and list pages containing pointers to entries of a particular category (e.g., a list of folk musicians). Strube and Ponzetto [2006]; Ponzetto and Strube [2006] make use of the category tree for computing semantic relatedness measures for the task. [Ratinov and Roth, 2012] utilize the category label directly

to derive a semantic class feature, along with other attributes (e.g., nationality, gender, etc.). Uryupina et al. [2011] link mentions to Wikipedia pages for the purpose of extracting *alias* information.

Yago [Suchanek et al., 2007] is an ontology combining information from Wikipedia and WordNet. It contains more than 1 million entities organised into a hierarchy, and 5 million facts about them (e.g., AlbertEinstein *bornInYear* 1879). The *type* and *means* relations have been utilized for coreference resolution [Bryl et al., 2010b,a; Uryupina et al., 2011; Rahman and Ng, 2011]. The former denotes the class an entity belongs to (e.g., AlbertEinstein *type* physicist), while the latter constitutes a mapping from a name to all of the entities it might refer to. For instance, "Einstein" might point to the the physicist Albert Einstein ("Einstein" *means* AlbertEinstein) or the musicologist Alfred Einstein ("Einstein" *means* AlfredEinstein).

Similarly to WordNet, the direct application of Wikipedia and related resources to coreference resolution is not always possible due to the presence of ambiguity. A mechanism for establishing a reliable link between a referring expression such as *Einstein* and its corresponding Wikipedia entry is necessary in order to fully take advantage of the category tree or the information stored in the info-box associated with this entity. This issue similarly affects the application of ontological data bases. Two referring expressions could be considered aliases of each other if at least one Yago *means* relation is found which holds between them. One cannot, however, reliably use the *type* relation to determine the semantic class of an entity without disambiguating it first. Bryl et al. [2010b] propose to address this problem in a word sense disambiguation step. The training data and sense inventory for this is provided by Wikipedia, thus allowing for a mention to be directly linked to its Wikipedia article[6] upon disambiguation. Another possibility is to use a dedicated entity linking tool [Uryupina et al., 2011; Ratinov and Roth, 2012].

A number of works have proposed to explore unannotated data as a source of semantic knowledge for coreference resolution. A widely-used strategy involves determining semantic relatedness of two expressions through lexico-syntactic patterns [Bean and Riloff, 2004; Daumé III and Marcu, 2005; Garera and Yarowsky, 2006; Ng, 2007; Yang and Su, 2007; Haghighi and Klein, 2009; Bansal and Klein, 2012]. Examples of such pattern include "$\langle \text{NP}_x \rangle$ *such as* $\langle \text{NP}_y \rangle$", and "$\langle \text{NP}_y \rangle$

[6]Linking to Wikipedia consequently provides access to other structured resources derived from the encyclopedia, such as DBPedia and the Yago ontology.

and other $\langle \text{NP}_x \rangle$", which are likely to express the hyponymy relation between NP_y and NP_x [Hearst, 1992]. Semantic relatedness cues can also be derived from patterns matching predicate-nominative (e.g., "$\langle \text{NP}_y \rangle$ *is a* $\langle \text{NP}_x \rangle$") and appositive constructions (e.g., "$\langle \text{NP}_y \rangle$, $\langle \text{NP}_x, \rangle$," as in $\langle$*Einstein*$\rangle$, $\langle$*a musicologist from Munich*$\rangle$,) [Daumé III and Marcu, 2005; Ng, 2007; Haghighi and Klein, 2009], or name–nominal pairs ($\langle$*musicologist*$\rangle$ $\langle$*Alfred Einstein*$\rangle$) [Fleischman et al., 2003; Rahman and Ng, 2011].

The patterns can be defined manually, or learned automatically. Yang and Su [2007] use the set of coreferential expressions from the training data as seeds, and find occurrences of these expressions in Wikipedia. The context words between the matched expressions are extracted as pattern candidates, which are subsequently filtered based on their frequency and association with other seed pairs. Ng [2007]; Rahman and Ng [2011] propose to learn such patterns directly from the coreference annotated corpus.

The obtained patterns can be utilized for coreference in several ways. They can be applied to a large unannotated data set in order to extract semantically-related pairs of nouns to serve as a knowledge base for the task [Fleischman et al., 2003; Daumé III and Marcu, 2005; Haghighi and Klein, 2009]. Alternatively, they can be used on the fly: if a pair of candidate co-referring expressions match one or more patterns in the current text or in a large unannotated corpus, this can be considered an indicator that they are semantically related. Bansal and Klein [2012] utilize Hearst patterns [Hearst, 1992] in order to obtain in-context co-occurrence counts of the head words of two mentions from the Google n-gram corpus [Brants and Franz., 2006].

Selectional preferences learned automatically from unannotated data have also been explored for the task, in particular for pronoun resolution [Hobbs, 1986; Dagan and Itai, 1990; Bergsma et al., 2008; Versley Y., 2016; Heinzerling et al., 2017]. The learning process involves collecting co-occurrence statistics of predicate–argument pairs, such as ⟨dog *subj* bite⟩ or ⟨drink *obj* milk⟩ from a large corpus. The extracted information can be further generalized by abstracting over word classes (e.g., ⟨animal *subj* bite⟩, ⟨ingest *obj* liquid⟩) [Resnik, 1993; Agirre and Martinez, 2001]. The learned preferences can be used to disambiguate between plausible antecedents of a referring expression by estimating how likely each antecedent is to fill the predicate role taken on by the referring expression in its local context. This

can be particularly beneficial for the resolution of short or semantically empty referring expressions, such as pronouns, where other commonly used features might fail to produce sufficient discriminative information.

Another related source of knowledge for the task is the output of a semantic parser [Ponzetto and Strube, 2006; Rahman and Ng, 2011]. Ponzetto and Strube [2006] identify verb predicates together and their arguments using a PropBank-style semantic role labelling tool, and include the argument role and predicate of each mention's head word as features. [Rahman and Ng, 2011] additionally introduce a feature indicating whether the predicates of the two mentions are part of the same FrameNet semantic frame.

The use of semantic features for coreference resolution has lead to mixed results. The comparison and analysis of the contribution of different types of semantic features is challenging due to mismatches in the reported evaluation metrics and the used corpora, as well as differences in the implementation and baseline set of features utilized in the experiments.

Some works report significant improvements in the coreference evaluation metrics (Section 5.3) as a result of introducing semantic and world knowledge to a baseline coreference resolver [Ponzetto and Strube, 2006; Ng, 2007; Rahman and Ng, 2011; Bansal and Klein, 2012; Recasens et al., 2013]. Rahman and Ng [2011] compare the effect of several semantic features extracted from YAGO, FrameNet, annotated coreference data and unstructured text. All features lead to improvements in B^3 and $CEAF$ scores on both the ACE'04 and OntoNotes corpora. While the gains brought by the features in isolation were modest, a combination of them lead to a substantial increase of approx. 3% for each score and data set. Recasens et al. [2013] extract coreferential pairs from comparable corpora, and use them as a source of semantic knowledge for the task. Their approach lead to 1% improvement on the OntoNotes corpus according to six different evaluation metrics, including the CoNLL F1 score. Bansal and Klein [2012] report improvements on the ACE'04 and ACE'05 data set as a result of their semantic web n-gram features (1.6% MUC, 1.2% B^3 and 2.1% MUC, 1.9% B^3 for ACE'04 and ACE'05, respectively).

Others works report little to no improvement over a semantically-poor baseline model [Uryupina et al., 2011; Durrett and Klein, 2013; Heinzerling et al., 2017]. Uryupina et al. [2011]'s semantic compatibility and aliasing features from Wikipedia and Yago brought no improvement over the baseline until a more complex

feature design strategy involving disambiguation and pruning of the knowledge resources was introduced. The updated feature extraction lead to an improvement of 1.6% MUC and CEAF (ACE'02). Heinzerling et al. [2017]'s selectional preference feature lead to only a minor improvement in CoNLL F1 (OntoNotes). Durrett and Klein [2013] utilize a number of semantic features including WordNet hypernymy and synonymy and clustering information, resulting in an improvement of 0.36% CoNLL F1 (OntoNotes) when all semantic features were included. Interestingly, the same set of features brought significant improvements when gold coreference mention spans were provided to the coreference system. While this greatly simplifies the task of automatic coreference resolution [Versley Y., 2016], it indicates that the provided semantic information was not a strong enough factor for establishing coreference links between the the large number of candidate mentions extracted by the coreference system.

The incorporation of semantic information to coreference resolution, while necessary, remains an uphill battle [Durrett and Klein, 2013]. The performance gains brought on by semantic features are limited due to the difficulty in accurately computing them given the available language technologies [Ng, 2007], as well as limitations in the coverage of the available resources (e.g., WordNet). To take full advantage of the available lexical semantic and encyclopedic knowledge, it is necessary to perform some form of disambiguation of mentions [Versley Y., 2016]. Entity linking is necessary in order to utilize Wikipedia and related knowledge bases for the task [Bryl et al., 2010b; Uryupina et al., 2011; Ratinov and Roth, 2012], while word sense disambiguation is needed for extracting reliable semantic information for common nouns from WordNet.

5.6 Word Embeddings as a Source of Semantic Knowledge

The creation of structured semantic resources is a costly and time-consuming process requiring expert knowledge. As a result, it is unsurprising that existing knowledge bases such as WordNet have been shown to exhibit certain limitations in coverage and are only available for a select number of languages.

Due to their ability to capture semantic and syntactic properties of words, word embeddings (Section 2.3) have become an essential building block in NLP applications. They carry the advantage of being obtained in an unsupervised manner, making them an accessible resource. While a typical use-case for word embeddings is representing textual input to deep learning models, the question arises whether and to what extend they can also serve as an alternative to structured resources for extracting semantic features for downstream tasks.

The methods used for evaluating the quality of word representations can offer some insights into the types of semantic information captured by them. Schnabel et al. [2015] describe a good word vector representation as one in which the relationship between two vectors reflects the linguistic relationships between the words they represent. *Intrinsic* evaluation techniques aim to measure this desired property directly using various data sets containing semantically-related words. Popular evaluation benchmarks include predicting semantic relatedness, synonymy, selectional preferences, analogy, and clustering concepts [Baroni et al., 2014].

Semantic relatedness data sets (e.g., WordSim353 [Finkelstein et al., 2001]) contain human judgements quantifying the degree to which the relatedness or similarity between pairs of words holds. *Relatedness* refers to the lexical semantic relations meronymy/ holonymy and broader topical similarity (e.g., student – school), while *similarity* encompasses synonymy/antonymy, hyponymy/hypernymy, and co-hyponymy (e.g., king – queen) [Baroni et al., 2014; Agirre et al., 2009]. The quality of the embeddings is judged based on the correlation of the human rating and the cosine similarity between the vector representations of the word pairs.

A common synonym detection data set is the TOEFL corpus [Landauer and Dumais, 1997]. Instead of ratings, this data set contains multiple-choice questions offering four synonym candidates per target term to choose from. The solution can be provided by selecting the answer with the highest cosine similarity to the target.

Concept categorization utilizes data sets of concepts organized into categories, such as *vehicles* or *mammals* [Baroni et al., 2014]. The evaluation process involves clustering of the vector representations of the given concepts into a predefined number of categories, and measuring the purity of the resulting clusters according to the gold standard groupings.

Selectional preference data sets contain verb–noun pairs, together with human judgements on how likely the noun is to serve as a subject or object of that verb. For instance, ⟨people *subj* eat⟩ would receive a high rating, while ⟨eat *obj* people⟩ would be ranked low. The task involves utilizing word representations to predict the selectional preference scores associated with the pairs.

The analogy task involves answering questions such as "*man* is to *father* as *woman* is to ⟨?⟩" (semantic analogy), or "*acceptable* is to *unacceptable* as *likely* is to ⟨?⟩" (syntactic analogy) [Mikolov et al., 2013a]. Mikolov et al. [2013a] propose to solve analogy via simple algebraic operations on word vector representations. Given an example pair of words which exhibit a certain relationship (father, man), and a query word (woman), the word which stands in a similar relationship to the query needs to be provided (mother). Subtracting the representation of "man" from the representation of "father" and adding the representation of "woman" should result in a vector which is close to that of "mother".

In order to gain an insight into how well predictive word embeddings (Section 2.3.2) fare in each of the tasks, we provide a brief summary of the findings of [Baroni et al., 2014]. Baroni et al. [2014] experimented with word2vec CBOW models with different hyper-parameters settings (e.g., context window size, dimensionality, optimization approach, etc.). On average across three data sets, the word relatedness task was solved with a correlation score of 80. The best achieved scores on the WordSim353 similarity and relatedness subsets were 80 and 70 Spearman correlation, respectively. On the synonymy prediction task (TOEFL), the best model achieved accuracy of 91%. The purity of the clusters obtained for concept categorization was 75%, 86%, and 99% on three separate data sets. Selectional preference prediction proved more challenging, with the best models achieving 28 and 41 Spearman correlation on two data sets. Finally, analogy prediction was evaluated on three data sets, resulting in accuracy values of 68, 71, and 66.

The results reveal that word embedding achieve reasonable precision in predicting semantic relations and categorizing concepts. This type of semantic information can prove valuable in a number of NLP tasks, including coreference resolution.

Previous works aimed to determine the usefulness of word embedding-based features for NLP tasks have focused on NER and chunking [Turian et al., 2010; Yu et al., 2013; Guo et al., 2014].

Turian et al. [2010] utilize word embeddings as additional token-level features for NER and chunking. The authors compare the unsupervised word representation models of [Collobert and Weston, 2008] and [Mnih and Hinton, 2008] against Brown clusters [Brown et al., 1992]. The former are included as multiple numerical features, where each dimension represents one feature. The latter constitute cluster labels of tokens, and are obtained via hierarchical clustering maximizing the mutual information of token bi-grams. Turian et al. [2010] found that each of the three sets of features lead to improvements over the baseline systems, with Brown clusters faring better compared to the other alternatives, in particular in the presence of rare words. Combining the three types of features lead to cumulative improvements for both tasks.

Yu et al. [2013] explore an alternative way of employing word embeddings for the two tasks at a lower computational cost. They argue that discrete word embedding-based features are better suited for use with linear models, as they result in better linear separability of the prediction classes. The proposed alternative is obtained by performing k-means clustering of vector representations, and assigning the resulting cluster labels as token-level features. More specifically, they utilize *compound* cluster features, described as conjunctive features of neighboring words. The underlying vector representations were obtained by the model of [Bengio et al., 2003]. The compound cluster features achieved better performance compared to Brown clusters and to using the word representations as numeric features, at much lower computational cost compared to the latter.

In [Guo et al., 2014], word embedding-based features are explored for the task of NER. The authors propose several different ways of deriving features from vector representations: 1) by creating a binarized version of the embedding to consider features with strong opinions on each dimension, 2) by clustering of the embedding via k-means and including the cluster label as a feature, and 3) as a distributional prototype feature. In the latter, a few prototypical examples are automatically selected to represent each type of named entity (Person, Organization, etc.). The prototype feature is activated if the cosine similarity between the embeddings of the word in question and any of the prototypes for a specific class is larger than a predefined threshold. The word representations utilized in the experiments were obtained with the skip-gram word2vec model. All word embedding-based features brought improvements to the baseline, and over the direct use of the embedding,

with the best performance achieved by the prototype feature. Moreover, a combination of the features led to a greater improvement in performance, indicating that the knowledge represented by them is not overlapping completely.

The work presented in the following chapter investigates the usefulness of word embedding-based features for supervised coreference resolution.

Chapter 6

Word Embeddings as Features for Supervised Coreference Resolution

6.1 Introduction

Coreference resolution involves automatically identifying and grouping expressions which refer to the same real-world entities (Section 5.1.2). It can provide valuable information to downstream NLP tasks which rely on language understanding, such as machine translation, text summarization, and question answering. Information extraction can also greatly benefit from coreference information. Grouping different mentions of the same entity can paint a more complete picture of the discourse participants involved in events and relations, and establish links between mentions of the same events through their arguments, aiding event linking.

Resolving reference is a non-trivial task (see Section 5.5.2). A referring expression often has multiple competing candidate antecedents. These may exhibit similar surface characteristics, such as agreement in number and gender. If the referring expression in question is semantically-poor (e.g., a pronoun), its successful disambiguation relies on exploring its sentential context.

Some form of semantic or world knowledge is often necessary in order to reliably determine the compatibility between different mentions of the same entity. Co-referring expressions might be linked via lexical semantic relations such as synonymy and hyperonymy (e.g., *a capital* is *a city*), world knowledge (e.g., *Paris* is

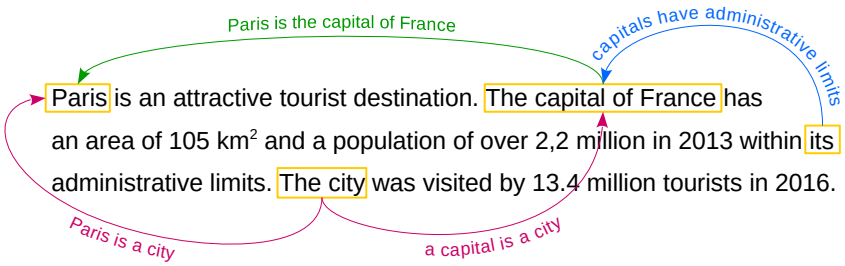

FIGURE 6.1: Lexical and encyclopedic knowledge can be helpful to determine the semantic compatibility between co-referring entity mentions: synonyms, hypernyms/hyponyms, world knowledge, referential attributes

the capital of *France*), and even attributive knowledge (e.g., *cities* have *areas* and *population*) (Figure 6.1).

A long line of work has been devoted to encoding semantic and encyclopedic information in the form of features for supervised coreference resolution (Section 5.5.2). Some of the investigated sources of such information include the lexical database WordNet [Daumé III and Marcu, 2005; Ponzetto and Strube, 2006; Bengtson and Roth, 2008], Wikipedia and related encyclopedic data bases [Strube and Ponzetto, 2006; Ratinov and Roth, 2012; Rahman and Ng, 2011] and mining unstructured data [Haghighi and Klein, 2009; Bansal and Klein, 2012]. Selectional preferences and semantic parsing have been employed for pronoun resolution [Ponzetto and Strube, 2006; Rahman and Ng, 2011; Heinzerling et al., 2017]. The experiments presented in this chapter investigate an alternative source of semantic knowledge for coreference resolution, namely distributed word representations.

Word embeddings are continuous, low-dimensional vector representations of words learned from large unlabeled corpora (Section 2.3). Predictive word representations are obtained as a byproduct of training a probabilistic language model. Similarly to distributional semantics approaches centered around the idea that the sense of a word can be defined by the company it keeps, the word embedding of a token is learned from its context words. As a result, similar representations are assigned to words which appear in similar contexts. Intrinsic evaluation has shown that word embeddings capture syntactic (e.g., gender) and semantic (e.g., relatedness and similarity) properties of words (Section 5.6). Being obtained in an unsupervised manner, they could also help address data sparsity issues in labeled training data at a small cost. For these reasons, word embeddings are an attractive source of semantic knowledge which be beneficial in multiple ways in a supervised learning task.

In this chapter we investigate whether and to what extend features derived from word embeddings can aid supervised coreference resolution. We experiment with several general purpose word embedding models, and several different types of embedding-based features, including embedding cluster and cosine similarity-based features. The inclusion of semantic features lead to improvements in the performance of a supervised coreference system. The work discussed in the rest of this chapter was published in [Simova and Uszkoreit, 2017].

6.2 Features Derived from Word Embeddings

Previous work has explored different ways of utilizing word embeddings as features for supervised learning NLP tasks (Section 5.6). Similarly to [Yu et al., 2013; Guo et al., 2014], we investigate the usefulness of a word embedding cluster label feature for coreference resolution (Section 6.2.1). Besides directly including the word embedding in the form of multiple numerical features as in [Turian et al., 2010], we experiment with more compact representations of it obtained through Principal Component Analysis (PCA) (Section 6.2.2). Finally, we define cosine similarity-based features, targeted at incorporating more of the context of the referring expression into the decision process (Section 6.2.3).

6.2.1 Embedding Cluster

The embedding cluster feature can be thought of as an approximation of the commonly-employed *semantic class* feature (Section 5.5.2). The latter is typically computed by traversing WordNet's taxonomy or Wikipedia's category tree. A semantic class feature can be beneficial for coreference resolution in that it can ensure that two candidate co-referring mentions are semantically compatible. For instance, a mention denoting a *vehicle* would not be compatible with a referent of type *person*.

We perform clustering of word embeddings with spherical k-means[1] [Dhillon and Modha, 2001]. Spherical k-means is similar to standard k-means, but utilizes *cosine similarity* (Equation (6.1)) as a distance metric, rendering it a suitable choice for comparing word embeddings. During clustering, each word embedding

[1] https://github.com/clara-labs/spherecluster

vector is treated as a single instance. The resulting clusters contain the words with the most similar embeddings according to the distance metric. We experiment with several different number of clusters in order to determine what granularity is most suitable for the task. The values include 50, 100, 500 for less fine-grained clusters, and 1k to 10k in steps of 2.5k, for more fine-grained ones.

The cluster labels are utilized for coreference in the following way: each mention's head word is equipped with a categorical feature denoting the label of the cluster this token was assigned to. Unsupervised cluster information has been previously utilized for coreference in [Durrett et al., 2013; Durrett and Klein, 2013]. To the best of our knowledge, this work is the first to explore word embedding-based clusters for the task.

We expect the embedding cluster feature to be useful in several ways. Coarse-grained cluster labels can directly ensure the semantic compatibility between different mentions of the same entity. The coreference resolver can learn compatible and incompatible anaphor–antecedent cluster combinations from the training data in the case of fine-grained cluster labels. We expect this feature to help improve generalization and thus tackle data sparsity issues of the training data. A drawback of utilizing word embeddings in this way is that each token is assigned to a single cluster. Word vectors encode different degrees of similarities between related words [Mikolov et al., 2013a], a property which is not exploited in this setup.

6.2.2 Dense Embedding Features

A straightforward way of utilizing word embeddings for coreference resolution is by directly including the embedding vector. In this setting, each dimension of the vector of a mention's head word is a separate numeric feature. The motivation behind this is that each dimension can be considered as a separate latent feature, encoding different syntactic or semantic properties of a word [Turian et al., 2010]. The model can learn which of these dimensions carry relevant information for the task.

One consideration here is that by including the whole vector we increase the amount of features substantially. Therefore we also experimented with more compact versions of the original vectors obtained through PCA[2]. PCA examines the

[2]package sklearn.decomposition.PCA

correlation between different dimensions in the word embedding vector, and identifies a smaller number of variables that preserve the variation in the original vector as best as possible. The vectors were reduced to retain different amounts of variability present in the original data, which may address over-fitting and lead to lower computational cost associated with training the model. We experiment with variance levels of 10% to 100% in steps of 10%.

6.2.3 Cosine Similarity Features

The third set of word embedding features is based on cosine similarity. Cosine similarity computes the cosine of the angle between two vectors q and d (Equation (6.1)). In the context of vector representations, cosine similarity has been used as a measure of the similarity and relatedness between words (Section 5.6).

$$cos(q, d) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} d_i^2}} \tag{6.1}$$

For each pair of candidate co-referring expressions, we compute four cosine similarity based features. We estimate the similarity between the head words of the anaphor (ANA) and antecedent (ANTE), as well as between their governing words (GOV_{ana}, GOV_{ante}) extracted from the dependency parse of the sentence, and the combinations ANA–GOV_{ante} and ANTE–GOV_{ana}.

The first similarity measure is meant to encourage the semantic compatibility between two mentions of the same entity, similarly to the embedding cluster feature. It is, however, a numeric feature rather than a categorical one.

As GOV_{ana} and GOV_{ante} are usually verbs, the second measure could prove useful in determining whether coreferring mentions carry out related actions (e.g., "During an interview he *said* [..] He further *reported* that [..]"). It can be thought of an approximation of the semantic parsing feature utilized in [Rahman and Ng, 2011], which indicates whether the predicates of the head words of two mentions are part of the same FrameNet semantic frame.

The last two similarity measures are meant as an approximation of selectional preferences. This features could prove useful in cases of pronoun resolution, where the context of the pronoun referring expression is a deciding factor. Consider the

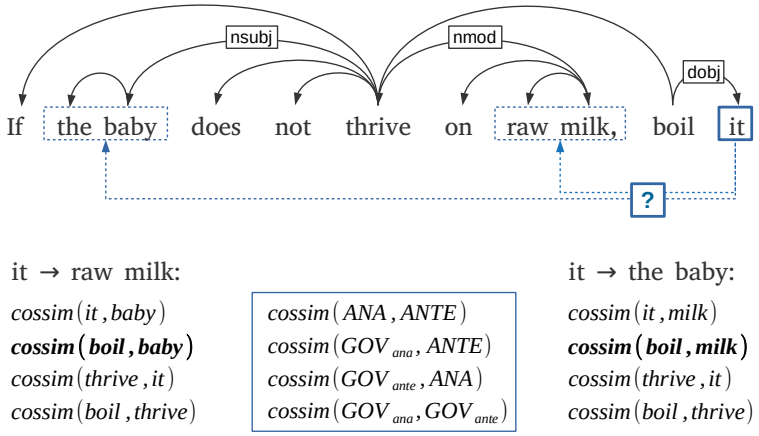

FIGURE 6.2: An example of the cosine similarity (*cossim*) features which would be used in determining the antecedent of the pronoun "it".

example from [Jespersen, 1949], "If the *baby* does not thrive on *raw milk*, boil *it*.", where the pronoun *it* has two candidate antecedents: *the baby* and *raw milk* (Figure 6.2). The cosine similarity between the governing word of the anaphor, *boil*, and the second antecedent candidate, *milk*, is very high, as opposed to the alternative pairing which leads to an undesirable sentence interpretation.

When calculating the cosine similarity between two tokens, we experimented with the original word embedding vectors, as well as their PCA-reduced versions.

6.3 Experimental Setup

This section provides more details on the setup of the experiments investigating the usefulness of the word embedding-based features. Section 6.3.1 presents the coreference resolution toolkit used throughout the experiments. Section 6.3.2 discusses the selected word embedding models and data sets for training and testing the coreference models. Section 6.3.3 provides details on the error analysis and evaluation methodology.

6.3.1 Coreference Resolver

The coreference resolution toolkit *cort*[3] [Martschat and Strube, 2015; Martschat et al., 2015] was used to obtain the baseline and extended models throughout our

[3] https://github.com/smartschat/cort

Feature	Description
fine_type(m)	name, definite/indefinite NP, demonstrative, verb, etc.
gender(m)	masculine, feminine, neuter, plural
number(m)	singular, plural
semantic_class(m)	person, object, numeric, unknown
deprel(m)	relation between NP head of m and dependency head of m
head_ner(m)	named entity tag of NP head of m
length(m)	num. tokens spanning m
head(m)	NP head of m
first(m)	first token of m
last(m)	last token of m
preceding_token(m)	token preceding m
next_token(m)	token following m
governor(m)	dependency head of the NP head of m
ancestry(m)	dep. head and grandparent POS tags and arc directions
exact_match(m_1, m_2)	m_1 and m_2 are exact string matches
head_match(m_1, m_2)	NP heads of m_1 and m_2 are exact string matches
same_speaker(m_1, m_2)	m_1 and m_2's speakers match (indirect speech))
alias(m_1, m_2)	m_1 is an alias of m_2 or vice versa
sentence_distance(m_1, m_2)	distance between m_1 and m_2 (num. sentences)
embedding(m_1, m_2)	m_1 spans over m_2 or vice versa
modifier(m_1, m_2)	agreement between modifiers of m_1 and m_2
tokens_contained(m_1, m_2)	m_1 is a substring of m_2 or vice versa
head_contained(m_1, m_2)	NP head of m_1 is a substring of NP head of m_2 or vice versa
token_distance(m_1, m_2)	distance between m_1 and m_2 (num. tokens)

TABLE 6.1: Coreference resolution features employed in cort [Martschat and Strube, 2015, 2014]. The *mention-pair* model operates on a pair of mentions m_1 and m_2, combining unary features of each mention, and pairwise features defined over both. For more information see https://github.com/smartschat/cort.

experiments. The system offers implementations of popular coreference resolution architectures, as well as a module for visualizing and comparing coreference resolution outputs. In this work we employ a *mention-pair* model with *best-first* clustering (Section 5.4.1). This model breaks down the task of coreference resolution into pairwise coreference decisions. The construction of coreference chains is subsequently enabled via the clustering step. The parameters of the model are learned via a structured latent perceptron with cost-augmented inference and averaging [Martschat and Strube, 2015]. Candidate coreference mention boundaries are determined automatically.

The baseline employs a standard set of features, including: number, gender, various string matching and distance features, fine type (name, definite/indefinite noun phrase, etc.), and others (see Table 6.1 for a complete list). The baseline set of features include a semantic class, which can take on the values "person", "object", or "numeric". As this class is very coarse grained, we chose to keep it and incorporate the word embedding-based features features alongside it. Section 6.5 offers a comparison of the effect of this feature and the embedding cluster feature on the model performance.

6.3.2 Data

The coreference data set used throughout our experiments is the OntoNotes corpus [Hovy et al., 2006; Weischedel et al., 2011]. We use the data splits employed in the CoNLL 2011 and 2012 shared tasks on modelling unrestricted coreference [Pradhan et al., 2011, 2012]. Following previous work [Martschat and Strube, 2015], all of our models were trained and tested on the version of the corpus with automatic preprocessing, mimicking a real-world coreference prediction scenario.

There is a fairly big selection of available pre-trained general purpose word embedding models. As a first step towards designing word embedding-based features, we selected three widely used models with varying properties:

- *w2v* – word2vec skip-gram with negative sampling embeddings[4] [Mikolov et al., 2013a]. These representations were trained on part of the Google News dataset, and have a vocabulary of 3 million words and phrases (e.g., New_York), and dimension of 300.
- *glove* – the GloVe embeddings[5] [Pennington et al., 2014] were trained on Wikipedia and Gigaword 5 [Parker et al., 2011], and have a vocabulary of 400K words, and dimension of 300.
- *deps* - the dependency-based word embeddings[6] [Levy and Goldberg, 2014] trained on part of Wikipedia, with a vocabulary size of about 180K words, and dimension of 300.

[4] https://code.google.com/archive/p/word2vec/
[5] https://nlp.stanford.edu/projects/glove/
[6] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

Our aim is not to provide a direct comparison of these models, but rather investigate whether some of them with their specific characteristics would prove particularly useful for the task. For instance, the *w2v* model is trained on data from the news domain, which coincides with the majority of the data in the OntoNotes coreference corpus, and has the largest vocabulary of the three. This model is also the only one to include representations of multi-word expressions. The *glove* model incorporates the encyclopedic data of Wikipedia in addition to the news data present in Gigaword. The *deps* embeddings differs from the previous two in that they takes a different perspective on which words in the context of a word are important for determining its sense. The model makes use of dependency analysis to select meaningful contexts. The resulting embeddings have been shown to exhibit more functional similarity [Levy and Goldberg, 2014], which can be beneficial for coreference resolution.

6.3.3 Evaluation and Analysis

The CoNLL F1 score, average of MUC, B^3, and $CEAF_e$ (see Section 5.3), is the main metric used to judge the quality of the models in the automatic evaluation. A paired bootstrap test[7] was selected for determining whether the improvements in the evaluation scores are statistically significant compared to the baseline, in accordance with the guidelines provided in [Dror et al., 2018]. We compare the evaluation scores assigned to each document section in the test corpus. The evaluation metrics were computed using the official coreference scorer[8] [Pradhan et al., 2014] of the CoNLL-2011/2012 task.

In addition, we perform error analysis of the results, using the methodology proposed in [Martschat and Strube, 2014]. In this study, coreference chains are viewed as complete one directional graphs, following the order in which mentions occur in the text. Error analysis is performed by comparing the system and gold-standard graphs in terms of their edges. A system's output is transformed into a maximum spanning tree, using a notion from accessibility theory [Ariel, 1990] to select the most likely missing links needed to reach the reference graph. If an edge in the resulting graph is missing from the reference entity, this is considered a recall error. The precision errors are extracted in analogous way. The advantage of this

[7] The test was carried out with the function `boot.t.test()` of the `R` library `MKinfer` (v0.5), see: https://rdrr.io/cran/MKinfer/man/boot.t.test.html.

[8] https://github.com/conll/reference-coreference-scorers (v8.01).

Feature Set			**F1**$_{\text{CONLL}}$
baseline			59.19
+EC$_{w2v,2.5k}$			59.56
+CS$_{glove,pca\text{-}20}$			59.49
+WE$_{glove,pca\text{-}50}$			59.67
+EC$_{glove,2.5k}$	+CS$_{glove,pca\text{-}20}$		59.68
+CS$_{deps,pca\text{-}60}$	+WE$_{glove,pca\text{-}50}$		59.66
+EC$_{glove,2.5k}$	+WE$_{glove,pca\text{-}50}$		59.66
+EC$_{glove,5k}$	+CS$_{deps,pca\text{-}90}$	+WE$_{glove}$	**59.72**†

TABLE 6.2: Automatic evaluation of the performance of the coreference system with different sets of features. The best model configurations were selected according to the CoNLL F1 score. The semantic features include: embedding cluster (EC) label, cosine similarity (CS), and word embedding (WE) (Section 6.2). Statistically significant improvements over the baseline are marked with †.

representation is that it allows for the recall and precision errors to be categorized in terms of type of anaphor and antecedent.

6.4 Results

6.4.1 Automatic Evaluation

The automatic evaluation results investigate the effect of each individual feature and combinations of them on the performance of the model. Several configurations of the embedding cluster (EC), word embedding (WE), and cosine similarity (CS) features were explored. These include three word embedding models (*w2v*, *glove*, *deps*), PCA-reduced versions of them retaining different amounts of variability of the original vectors (10%–100%), and different number of clusters determining the granularity of the EC feature (50, 100,500,1K, 2.5K, 5K, 7.5K,10K). Table 6.2 contains the best results achieved by the coreference resolution models for each feature and feature combination, according to the CoNLL F1 score.

The best configuration for the embedding cluster feature was the *w2v* embedding and 2.5K number of clusters. The same number of clusters lead to the best results in conjunction with the other two embedding models (not shown). We observed a worsening of the performance compared to the baseline when the number of clusters exceeded 7.5K, regardless of the choice of word embedding.

Feature Set	MUC	BCUB	CEAF$_m$	CEAF$_e$	BLANC	F1$_{\text{CONLL}}$
baseline	69.44	56.32	59.88	51.81	57.24	59.19
+EC	69.80	56.65	60.06	52.24	57.14	59.56
+CS	69.67	56.56	59.87	52.23	56.92	59.49
+WE	**69.91**†	56.80	60.08	52.29	57.35	59.67
+EC+CS+WE	69.90	**56.82**†	60.24	52.45	57.23	**59.72**†

TABLE 6.3: Detailed automatic evaluation results including various coreference metrics (see Section 5.3 for an overview). Statistically significant improvements over the baseline are marked with †.

For the cosine similarity set of features, the best performance was achieved by *glove* with explained variance of 20%. Using the original word embedding vectors to compute this feature lead to a drop in CoNLL F1 for *deps* and *glove* compared to the baseline, and small improvement for *w2v*.

The best individual feature contribution was achieved by the word embedding feature with *glove* and variance of 50%. Similarly to the cosine feature, we observed that when the original embeddings were used, the performance was worse compared to the PCA-reduced versions of the vectors.

The second part of Table 6.2 shows the best pairwise combinations of features. Our goal was to determine if the knowledge they encode is complementary. We did not observe any gains in performance over the use of the best individual feature. The best result was achieved by the combination of the embedding cluster and cosine similarity features.

Using the three features in conjunction lead to the best result of 59.72 F1. The improvement in score compared to the baseline, albeit modest, is statistically significant according to the bootstrapped paired t-test (p-value=0.036, mean of the differences=0.41). Utilizing the best individual feature configurations did not lead to the best combination of the three features.

A detailed summary of the automatic evaluation scores for the baseline model, individual feature contributions, and best combination, is presented in Table 6.3. The semantic features lead to better or comparable results according to all evaluation scores with the exception of BLANC, which decreased for the EC and CS models. The WE model achieved a statistically-signifficant improvement in MUC score, while the combination model resulted in a statistically better BCUB score in addition to the CoNLL F1 score.

Feature Set	**coref. links**			**non-coref. links**			**total**		
	P	R	F1	P	R	F1	P	R	F1
baseline	58.5	57.2	57.8	58.8	54.7	56.7	58.6	55.9	57.2
+EC	56.8	58.0	57.4	59.4	54.5	56.9	58.1	56.3	57.1
+CS	56.7	57.1	56.9	59.8	54.4	57.0	58.2	55.7	56.9
+WE	58.9	56.8	57.8	60.2	53.9	56.9	59.5	55.4	57.4
+EC+CS+WE	58.8	56.2	57.4	59.7	54.6	57.0	59.3	55.4	57.2

TABLE 6.4: Detailed BLANC precision, recall and F1 scores for coreferent and non-coreferent links and combined results.

Table 6.4 provides the detailed BLANC results for coreferent and non-coreferent links and the total scores. The EC and CS models lead to a drop in precision for coref. links, while the CS also lead to a drop in recall, which accounts for the worse overall performance of these models according to the average BLANC score. Both models lead to an improvement in precision for non-coref. links, however. The WE and combination models lead to an improvement in precision at the expense of recall for both coref. and non-coref. links.

6.4.2 Error Analysis

The error analysis presented in this section offers a better insight into where the word embedding feature models differ with respect to the baseline model and each other. We utilize the methodology proposed by [Martschat and Strube, 2014] to extract and categorize the errors made by each model.

Figure 6.3 presents the distribution of errors with respect to the type of referring expression. The figure shows the absolute number of precision and recall errors in the resolution of pronouns (PRO), named entities (NAM), nominals (NOM), and demonstratives (DEM). To put these numbers in perspective, the total number of nominal mentions in the test set is approx. 9.2K (43%), the number of pronouns is approx. 8.2K (39%), 2.3K (11%) are names, and about 500 (2%) are demonstratives.

All word embedding models lead to a reduction of errors in pronoun resolution compared to the baseline. The best performance was achieved by the combination model, which resulted in the lowest number of both precision and recall errors. The WE model was second best in terms of precision, and third with respect to

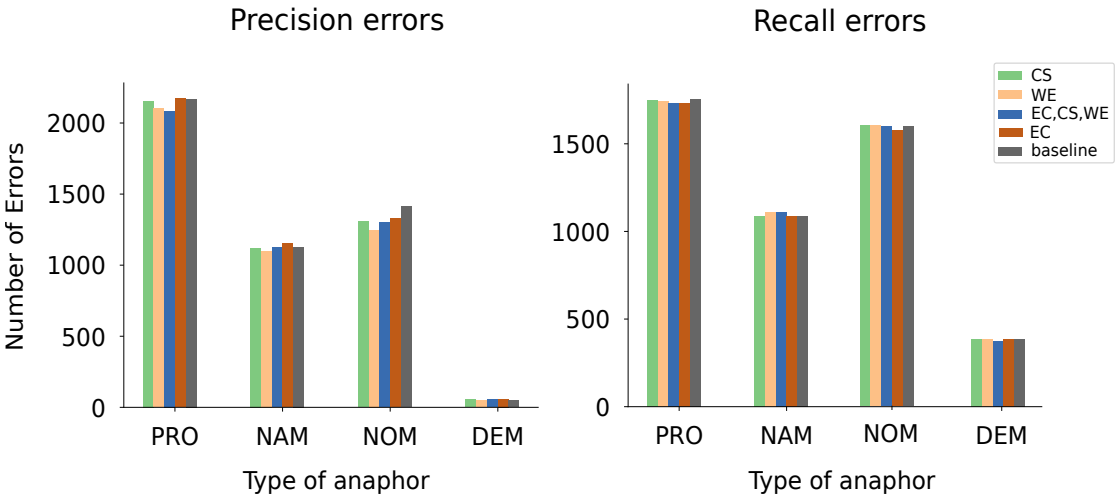

FIGURE 6.3: Distribution of precision and recall errors per type of referring expression: pronouns (PRO), named entities (NAM), nominals (NOM), and demonstratives (DEM).

recall. Despite being specifically intended for assisting pronoun resolution, the CS model landed third with respect to precision, and achieved comparable recall to the baseline system. Finally, the EC model lead to comparable precision to the baseline, but was second best in terms of recall.

The lowest number of precision errors in the resolution of names was achieved by the WE model, followed by the CS and combination models. The EC feature lead to a degradation in precision compared to the baseline. In terms of recall, the WE and combination systems were worse than the baseline, while the remaining models achieved comparable results.

All word embedding models achieved a reduction in the number of precision errors for the resolution of nominal mentions. The WE model was best in terms of precision, but lead to a degradation in recall compared to the baseline. The combination model was second best in both precision and recall. The EC model achieved the best recall of the five models.

The number of precision and recall errors in the resolution of demonstratives is comparable across models. The WE model was the best in terms of precision errors, while the combination model obtained the lowest number of recall errors.

Table 6.5 and Table 6.6 offer a more detailed breakdown of the errors made by each model. The values provided in each table represent the number of errors (raw counts) per type of anaphor–antecedent link, focusing on the three most common mention types: names, nominals, and pronouns. For instance, the baseline system

Precision Errors						
Error Type	**Feature Set**					
ANA⇒ANTE	*baseline*	+EC	+CS	+WE	+EC+CS+WE	COMMON
PRO⇒PRO	1565	**1554**	1566△	**1562**	**1520**	1103
PRO⇒NAM	217	231△	239△	**215**	**213**	124
PRO⇒NOM	346	**342**	**311**	**283**	**316**	166
NAM⇒PRO	70	79△	77△	**63**	**57**	28
NAM⇒NAM	997	999△	**982**	**984**	1001△	732
NAM⇒NOM	61	75△	**60**	**54**	65△	36
NOM⇒PRO	49	59△	54△	**31**	**43**	14
NOM⇒NAM	77	**61**	**71**	**69**	**71**	26
NOM⇒NOM	1285	**1205**	**1180**	**1148**	**1187**	785
TOTAL	4758	**4712**	**4638**	**4500**	**4565**	3014

TABLE 6.5: Precision error analysis. The values represent the number of errors (raw counts) per type of anaphor–antecedent link. A decrease in the amount of errors is marked in **bold**, and an increase: with the symbol △.

made 1565 precision errors when resolving a link between a pronoun anaphor and a pronoun antecedent (PRO⇒PRO) out of 4758 errors in total. In bold we mark all cases where a reduction of the amount of errors is visible when comparing to the baseline system, while an increase in the amount of errors is marked with the symbol △. The last column provides the number of errors common to all models.

All of the models achieved a reduction in the *total* number of precision errors compared to the baseline system. The word embedding model comes first with 258 fewer errors, followed by the best combination model with 193 fewer errors. In addition, all models improved in the following categories: NOM⇒NOM (best: WE, 137 fewer errors), NOM⇒NAM (best: EC, 16 fewer errors), and PRO⇒NOM (best: WE, 63 fewer errors). It is encouraging that all of the models improve over the baseline in two of the categories involving nominal anaphors, while the combination and WE models improve in all three, showcasing the influence of the semantic knowledge introduced by the word embedding features.

Our initial hypothesis that the cosine similarity model would be especially useful for pronoun resolution was not supported by the precision error analysis. For the category PRO⇒NAM, it even introduced 22 additional errors. The combination and WE models improved pronoun resolution for all categories.

Recall Errors						
Error Type	**Feature Set**					
ANA⇒ANTE	*baseline*	+EC	+CS	+WE	+EC+CS+WE	COMMON
PRO⇒PRO	348	348	358△	354△	348	279
PRO⇒NAM	475	**463**	**469**	**464**	**462**	394
PRO⇒NOM	824	**817**	**820**	**818**	**815**	702
NAM⇒PRO	63	**60**	**60**	65△	65△	54
NAM⇒NAM	850	857△	854△	864△	867△	800
NAM⇒NOM	169	**166**	**168**	172△	172△	160
NOM⇒PRO	27	**26**	28△	**26**	27	25
NOM⇒NAM	442	**429**	**440**	449△	**441**	406
NOM⇒NOM	1013	**1003**	1022△	1013	**1012**	919
TOTAL	4844	**4800**	4853△	4861△	**4835**	3739

TABLE 6.6: Recall error analysis. The values represent the number of errors (raw counts) per type of anaphor–antecedent link. A decrease in the amount of errors is marked in **bold**, and an increase: with the symbol △.

The WE model stands out in terms of precision compared to all other models, leading to a reduction of errors in each category.

While the EC model improved in some categories, it resulted in the highest total number of precision errors of all word embedding feature models. The EC model did, however, achieve the lowest number of recall errors (44 fewer), improving in all but two categories compared to the baseline (Table 6.6). The only other model which achieved a reduction in the total number of recall errors was the combination model (9 fewer). The reduction in precision errors achieved by the WE model came at the expense of recall, leading to an increase in the total number of recall errors by 17. All models improved recall in the categories PRO⇒NAM (best: combination model, 13 fewer errors), and PRO⇒NOM (best: combination model, 9 fewer errors).

In terms of common errors, 77% of the recall errors made by the baseline are shared with the remaining models, while only 63% of the precision errors are common to all. The categories with most common recall errors compared to the baseline are NAM⇒NAM (94%), and NAM⇒NOM (95%), while for precision they are PRO⇒PRO (71%), and NOM⇒NOM (61%). On the other hand, the categories with least overlap compared to the baseline were PRO⇒PRO (80%) and PRO⇒NAM (83%) for recall, and NOM⇒PRO (29%) and NOM⇒NAM (34%) for precision.

The errors analysis reveals that biggest challenge for all models was the resolution of names. The total number of name mentions in the test set is approx. 2300. Common to all models were 1020 recall and 796 precision errors associated with this type of expression. For comparison, there are approximately 8200 pronoun anaphors in the test set, while the total number of common errors were 1469 (recall), and 1412 (precision). With respect to the 9200 nominal mentions, the models made 1462 recall errors and 825 precision errors.

6.5 Discussion

On the topic of selecting a word embedding model, we made several observations. In the automatic evaluation and error analysis for most of our experiments, including ones not reported here, the *glove* word embedding proved to be most useful out of the three (Table 6.2). The best embedding cluster configuration was achieved with the *w2v* model, which could perhaps be attributed to the fact that it has the largest vocabulary of the three. The *deps* model stood out in conjunction with the cosine similarity feature. Given that it has the smallest vocabulary of the three, it would be interesting so see how this type of embedding performs for the task when trained on a bigger data set. This would also allow us to fairly compare the former models, encoding both topical and functional similarity, and the latter with more functional similarity [Levy and Goldberg, 2014].

The embedding cluster feature was the most cost-effective way of providing word embedding information to the coreference tool. Unlike the findings of previous works [Yu et al., 2013; Guo et al., 2014], however, it did not lead to better performance over the direct use of the word embedding. The word embedding feature did lead to longer training and testing times, but using the PCA reduction technique improved on both running time and F1 measure.

Additional complexity is added to the coreference model when attempting to combine several of the word embedding features. Moreover, it is not trivial to find a good combination - selecting the features configurations which perform best in isolation did not lead to the most successful combination.

We performed an additional study of the effect of the newly introduced word embedding features when some of the original features had been removed. The goal of this experiment was to investigate to what extend the knowledge encoded

Feature Set		F1	ΔF1
baseline		59.19	
+EC$_{w2v,2.5k}$		59.56	+0.37
+WE$_{glove,pca\text{-}50}$		59.67	+0.48
-gender		59.10	-0.09
-gender	+EC$_{w2v,2.5k}$	59.18	-0.01
-gender	+WE$_{glove,pca\text{-}50}$	59.34	+0.15
-number		59.39	+0.20
-number	+EC$_{w2v,2.5k}$	59.33	+0.14
-number	+WE$_{glove,pca\text{-}50}$	59.43	+0.24
-head		59.19	+0.00
-head	+EC$_{w2v,2.5k}$	59.28	+0.09
-head	+WE$_{glove,pca\text{-}50}$	59.29	+0.10
-head_ner		59.14	-0.05
-head_ner	+EC$_{w2v,2.5k}$	59.34	+0.15
-head_ner	+WE$_{glove,pca\text{-}50}$	59.34	+0.15
-semantic_class		59.23	+0.04
-semantic_class	+EC$_{w2v,2.5k}$	59.44	+0.25
-semantic_class	+WE$_{glove,pca\text{-}50}$	59.46	+0.27
-fine_type		55.50	-3.69
-fine_type	+EC$_{w2v,2.5k}$	55.98	-3.21
-fine_type	+WE$_{glove,pca\text{-}50}$	55.74	-3.45

TABLE 6.7: Interaction of some of the original and newly introduced features. ΔF1 denotes the difference in the performance of a model compared to the baseline.

by these features overlaps with some of the syntactic and semantic information already present in the original ones, and whether they might be better alternatives to them. These include: gender and number information, head word and Named Entity (NE) tag of the head word, and semantic class and fine type of the mention. Table 6.7 presents a summary of the results.

None of the new features seem to completely cover the information encoded by the original ones. Rather, they work best in the setting where they interact with each other.

The WE features encode more of the *number* and *gender* information, compared to the embedding cluster feature. This is to be expected as certain dimensions of the embeddings likely encode these properties, while the information is lost

after performing clustering. It was surprising that simply removing the *number* feature lead to an improvement in F1 score. This is possibly due to errors in the computation of this feature, but a more detailed analysis is needed to confirm this hypothesis.

Both of the newly introduced features perform similarly in the absence of the head word, NE tag, and semantic class features. The removal of the semantic class feature lead to a slight improvement in F1 score. This feature can be difficult to compute correctly, as it requires access to a knowledge base for extracting the category of common nouns. In the case of the fine type feature, the EC slightly outperforms the WE feature, but neither manage to cover much of the information encoded in the original feature.

6.6 Conclusion

This work offers some insights on how word embeddings can be applied to the task of coreference resolution. We present three different features derived from word embeddings, and show that they influence the coreference resolution process in different ways. These include a setting in which each dimension of the embedding is a separate numeric feature, an embedding cluster which approximates a semantic class, and a set of cosine similarity features, which incorporate some contextual information.

Our evaluation results and error analysis show that each of these features helps to improve over the baseline coreference system's performance. The improvement in CoNLL F1 score brought on by utilizing all of the proposed features in conjunction was statistically significant. We observed a reduction in the total number of precision errors. Moreover, all features lead to a reduction in the amount of precision errors in resolving references between *common nouns*. These results suggest that they successfully incorporate some semantic information into the process.

The word embedding models explored in this work have certain limitations with respect to their handling of out-of-vocabulary terms and polysemous words (Section 2.3). Furthermore, only one of them attempts to represent multiword units, which are common among named entity anaphors. These shortcomings can impact the usefulness of the models for the task, similarly to the way in which the lack of

word sense disambiguation has influenced the utilization of structured knowledge bases as a source of semantic knowledge in previous works (Section 5.5.2).

A plausible future direction for this experiment is the exploration of the more recently introduced class of word representations referred to as contextualized word embeddings (Section 2.3.3). Contextualized word embeddings are better adept at representing polysemous words, and mitigate the effect of out-of-vocabulary words via the use of sub-word information. As these representations are functions of the entire sentential context of a token, they are also a better choice in the presence of multi-word expressions.

In recent years, the previously widely-used models for CR with hand-crafted features have been largely replaced with deep neural network-based models (Section 5.4.2). While these models have lead to advancements in the resolution of coreference, their success is dependent on the availability of sufficient amounts of labelled training data, only available for English and a select number of languages. In lower-resource scenarios, models with hand-crafted features can be advantageous, as they encode much of the expert knowledge relevant for the task explicitly. Being obtained in an unsupervised manner, word embeddings can be an attractive and accessible source of semantic information for such models.

Chapter 7

Conclusion

This thesis presents research on two NLP tasks with relevance to the topic of cross-sentence Relation Extraction, namely *event extraction* and *coreference resolution*.

7.1 Thesis Summary

The first part of the thesis presents a semi-supervised method for obtaining labeled examples for the task of event extraction. The approach is based on self-training, and relies on the availability of a small number of labeled examples to train an initial model for event detection. This model is subsequently used to discover new event instances in a large collection of unstructured text. Only the high confidence model decisions are taken into account when selecting automatically-labeled examples to construct what we refer to as an augmentation data set. This data set can subsequently be used for re-training the event tagger, in conjunction with the original corpus.

We propose a strategy for utilizing the automatically extracted data alongside the existing gold-standard corpus which yield superior performance compared to simply concatenating the two data sets. The proposed approaches dampens the effect of the augmentation data through a scaling parameter applied to the loss during training of the model with the possibly noisy new examples. In the conclusion of Chapter 4 we outline several alternatives to this approach which would be interesting to investigate in future work.

Another contribution of this work is the comparison between different architectures for sequence labeling applied to event detection. We experimented with several deep learning architectures, including a Bi-LSTM, a Bi-LSTM with a CRF decoder, and a transformer. We also tested the effect of different contextualized and non-contextualized word embedding models for representing the input to the model.

The second part of the thesis focuses on the problem of coreference resolution. While certain level of performance in supervised CR can be reached with modeling surface information about entity mentions, their successful resolution often depends on semantic or world knowledge. A large amount of work has been devoted to the creation of semantic features from structured knowledge sources. This thesis investigates an unsupervised source of such knowledge, namely distributed word representations.

We present several ways in which word embeddings can be utilized to extract features for a supervised coreference resolver. These include: a clustering feature approximating a semantic class, numeric features obtained by reducing the dimensionality of the vector representation, and cosine similarity-based features which incorporate additional contextual information. We experimented with several widely-used non-contextualized word embedding models. We conclude with a detailed automatic evaluation and error analysis showcasing the contribution of the word embedding features.

7.2 Outlook

The central motivation behind the experiments described in this thesis was enabling the extraction of relation mentions which are expressed in larger context than a single sentence. We observed that the two main reasons for spreading of this information are event and entity coreference, and focused on the sub-problems of coreference resolution and event extraction.

While coreference resolution is a very well studied problem in NLP, it remains a challenging one. Around the time the work on this thesis began, the performance of state-of-the-art coreference resolvers was averaging 62% ConLL F1 score (OntoNotes), achieved by supervised models utilizing hand crafted features [Björkelund and Kuhn, 2014; Martschat and Strube, 2015]. The first deep learning

approaches introduced soon thereafter pushed this limit to about 65% [Wiseman et al., 2016; Clark and Manning, 2016a,b]. This performance is well below levels acceptable for real-world applications in a domain such as IE, where the aim is to extract factual information. This prompted our interest in the incorporation of semantic features for the task with the aim of improving the reliability of CR technology.

Recent advances in deep learning have lead to considerable improvements in the performance of CR models, reaching 80% CoNLL F1 score [Xu and Choi, 2020]. These advances bring the technology one step closer to its successful application in downstream tasks such as relation extraction, event extraction, and event linking, among others. We expect CR will remain an active area of research in the years to come. To the best of our knowledge, the question whether the lack of semantic information remains a problem in CR with the current deep learning approaches has not been substantially explored. Future work should investigate this to shed light on the issues which remain unsolved in CR.

We proposed a method for obtaining automatically-labeled examples of events through self-training. While self-training is not novel in NLP, it is under-investigated for event and relation extraction in comparison to other semi-supervised approaches, such as distant supervision. We consider self-training particularly well suited for the discovery of event instances and the discovery of cross-sentential relations because the process is not limited to extracting event mentions which contain a certain number of named entity arguments within the same sentence. In many cases, the arguments of an event may be referenced in a sentence containing event triggers, but be fully realized somewhere else in the document. Krause [2018] estimates that this affects 77% of the event mentions in ACE 2005. Such event mentions would not be detected by distant supervision unless a larger text span is considered, which would also increase the likelihood of false positives. Since self-training is not guided by arguments but rather by the event triggers observed in a small number of labeled examples, it is able to detect event mentions without imposing a restriction on the number or type of event participants mentioned in close proximity. A combination of a supervised model for event extraction and a reliable coreference resolver can put together the missing pieces of information found in cross-sentential mentions. Thus, self-training might be the better semi-supervised method for the discovery of cross-sentential relations in unstructured

data. We are not aware of any work focusing on cross-sentence RE which explores this approach, and would be interested to see it in future work.

Appendix A

Tables

Table A.1: ACE 2005 Data Set Split

test set		
nw	AFP_ENG_20030504.0248	AFP_ENG_20030617.0846
AFP_ENG_20030401.0476	AFP_ENG_20030508.0118	AFP_ENG_20030625.0057
AFP_ENG_20030413.0098	AFP_ENG_20030508.0357	AFP_ENG_20030630.0271
AFP_ENG_20030415.0734	AFP_ENG_20030509.0345	APW_ENG_20030304.0555
AFP_ENG_20030417.0004	AFP_ENG_20030514.0706	APW_ENG_20030306.0191
AFP_ENG_20030417.0307	AFP_ENG_20030519.0049	APW_ENG_20030308.0314
AFP_ENG_20030417.0764	AFP_ENG_20030519.0372	APW_ENG_20030310.0719
AFP_ENG_20030418.0556	AFP_ENG_20030522.0878	APW_ENG_20030311.0775
AFP_ENG_20030425.0408	AFP_ENG_20030527.0616	APW_ENG_20030318.0689
AFP_ENG_20030427.0118	AFP_ENG_20030528.0561	APW_ENG_20030319.0545
AFP_ENG_20030428.0720	AFP_ENG_20030530.0132	APW_ENG_20030322.0119
AFP_ENG_20030429.0007	AFP_ENG_20030601.0262	APW_ENG_20030324.0768
AFP_ENG_20030430.0075	AFP_ENG_20030607.0030	APW_ENG_20030325.0786
AFP_ENG_20030502.0614	AFP_ENG_20030616.0715	

dev set		
nw	CNN_IP_20030402.1600.00-1	**un**
AFP_ENG_20030304.0250	CNN_IP_20030405.1600.01-1	marcellapr_20050228.2219
AFP_ENG_20030305.0918	CNN_IP_20030409.1600.02	rec.games.chess.politics_20041216.1047
AFP_ENG_20030311.0491		rec.games.chess.politics_20041217.2111
AFP_ENG_20030314.0238	**bn**	soc.org.nonprofit_20050218.1902
AFP_ENG_20030319.0879	CNNHL_ENG_20030304_142751.10	
AFP_ENG_20030320.0722	CNNHL_ENG_20030424_123502.25	**wl**
AFP_ENG_20030327.0022	CNNHL_ENG_20030513_220910.32	FLOPPINGACES_20050217.1237.014
AFP_ENG_20030327.0224	CNN_ENG_20030304_173120.16	AGGRESSIVEVOICEDAILY_20041116.1347
	CNN_ENG_20030328_150609.10	FLOPPINGACES_20041117.2002.024
bc	CNN_ENG_20030424_070008.15	FLOPPINGACES_20050203.1953.038
CNN_CF_20030303.1900.02	CNN_ENG_20030512_170454.13	TTRACY_20050223.1049
CNN_IP_20030329.1600.00-2	CNN_ENG_20030620_085840.7	

train set		
nw	CNN_ENG_20030403_183513.1	CNNHL_ENG_20030523_221118.14
APW_ENG_20030326.0190	CNN_ENG_20030404_073033.4	CNNHL_ENG_20030526_221156.39
APW_ENG_20030327.0376	CNN_ENG_20030404_163526.10	CNNHL_ENG_20030603_230307.3
APW_ENG_20030331.0410	CNN_ENG_20030407_080037.12	CNNHL_ENG_20030604_230238.5
APW_ENG_20030403.0862	CNN_ENG_20030407_130604.10	CNNHL_ENG_20030609_133335.37

APW_ENG_20030404.0439
APW_ENG_20030406.0191
APW_ENG_20030407.0030
APW_ENG_20030408.0090
APW_ENG_20030409.0013
APW_ENG_20030410.0906
APW_ENG_20030411.0304
APW_ENG_20030412.0531
APW_ENG_20030414.0392
APW_ENG_20030415.0742
APW_ENG_20030416.0581
APW_ENG_20030417.0555
APW_ENG_20030418.0084
APW_ENG_20030419.0358
APW_ENG_20030422.0469
APW_ENG_20030422.0485
APW_ENG_20030423.0079
APW_ENG_20030424.0532
APW_ENG_20030424.0698
APW_ENG_20030502.0470
APW_ENG_20030502.0686
APW_ENG_20030508.0772
APW_ENG_20030510.0228
APW_ENG_20030513.0139
APW_ENG_20030519.0367
APW_ENG_20030519.0548
APW_ENG_20030520.0081
APW_ENG_20030520.0757
APW_ENG_20030527.0232
APW_ENG_20030602.0037
APW_ENG_20030603.0303
APW_ENG_20030610.0010
APW_ENG_20030610.0554
APW_ENG_20030619.0383
NYT_ENG_20030403.0008
NYT_ENG_20030602.0074
NYT_ENG_20030630.0079
XIN_ENG_20030314.0208
XIN_ENG_20030317.0177
XIN_ENG_20030324.0191
XIN_ENG_20030327.0202
XIN_ENG_20030408.0341
XIN_ENG_20030415.0379
XIN_ENG_20030423.0011
XIN_ENG_20030425.0184
XIN_ENG_20030509.0137
XIN_ENG_20030513.0002
XIN_ENG_20030523.0202
XIN_ENG_20030609.0118
XIN_ENG_20030610.0299
XIN_ENG_20030616.0274
XIN_ENG_20030624.0085
AFP_ENG_20030330.0211
AFP_ENG_20030323.0020

cts

fsh_29097
fsh_29105
fsh_29121
fsh_29138
fsh_29139
fsh_29141

CNN_ENG_20030407_170605.7
CNN_ENG_20030408_083034.11
CNN_ENG_20030408_123613.0
CNN_ENG_20030408_153616.9
CNN_ENG_20030408_200618.14
CNN_ENG_20030409_180633.8
CNN_ENG_20030410_183644.8
CNN_ENG_20030411_193701.3
CNN_ENG_20030411_233701.11
CNN_ENG_20030414_130735.7
CNN_ENG_20030415_103039.0
CNN_ENG_20030415_173752.0
CNN_ENG_20030415_180754.5
CNN_ENG_20030415_183752.14
CNN_ENG_20030416_100042.7
CNN_ENG_20030416_160804.4
CNN_ENG_20030416_180808.15
CNN_ENG_20030416_190806.4
CNN_ENG_20030417_063039.0
CNN_ENG_20030417_073039.2
CNN_ENG_20030418_063040.1
CNN_ENG_20030418_083040.11
CNN_ENG_20030418_130831.5
CNN_ENG_20030418_163834.14
CNN_ENG_20030421_090007.11
CNN_ENG_20030421_120508.13
CNN_ENG_20030421_120508.17
CNN_ENG_20030421_133510.6
CNN_ENG_20030422_083005.10
CNN_ENG_20030422_213527.4
CNN_ENG_20030423_180539.2
CNN_ENG_20030424_073006.4
CNN_ENG_20030424_113549.11
CNN_ENG_20030424_173553.8
CNN_ENG_20030424_183556.7
CNN_ENG_20030425_063006.5
CNN_ENG_20030425_133605.6
CNN_ENG_20030426_160621.0
CNN_ENG_20030428_130651.4
CNN_ENG_20030428_173654.13
CNN_ENG_20030428_193655.2
CNN_ENG_20030429_083016.5
CNN_ENG_20030429_110706.7
CNN_ENG_20030429_143706.14
CNN_ENG_20030429_170710.4
CNN_ENG_20030429_190711.14
CNN_ENG_20030430_063016.14
CNN_ENG_20030430_093016.0
CNN_ENG_20030430_160723.6
CNN_ENG_20030501_063017.15
CNN_ENG_20030501_160459.0
CNN_ENG_20030502_080020.7
CNN_ENG_20030502_093018.6
CNN_ENG_20030505_090022.1
CNN_ENG_20030506_053020.14
CNN_ENG_20030506_160524.18
CNN_ENG_20030506_163523.22
CNN_ENG_20030507_060023.1
CNN_ENG_20030507_160538.15
CNN_ENG_20030507_170539.0
CNN_ENG_20030508_170552.18
CNN_ENG_20030508_210555.5

CNNHL_ENG_20030610_133347.6
CNNHL_ENG_20030610_230438.14
CNNHL_ENG_20030611_133445.24
CNNHL_ENG_20030616_230155.28
CNNHL_ENG_20030616_230155.7
CNNHL_ENG_20030618_230303.36
CNNHL_ENG_20030618_230303.6
CNNHL_ENG_20030624_133331.33
CNNHL_ENG_20030624_230338.34
CNNHL_ENG_20030625_193346.7
CNNHL_ENG_20030625_230351.4
CNNHL_ENG_20030410_193626.13

wl

AGGRESSIVEVOICEDAILY_20041101.1144
AGGRESSIVEVOICEDAILY_20041101.1806
AGGRESSIVEVOICEDAILY_20041201.2313
AGGRESSIVEVOICEDAILY_20041203.1959
AGGRESSIVEVOICEDAILY_20041208.2133
AGGRESSIVEVOICEDAILY_20041215.2302
AGGRESSIVEVOICEDAILY_20041218.0146
AGGRESSIVEVOICEDAILY_20041218.1004
AGGRESSIVEVOICEDAILY_20041223.1449
AGGRESSIVEVOICEDAILY_20041226.1712
AGGRESSIVEVOICEDAILY_20050105.1344
AGGRESSIVEVOICEDAILY_20050106.1310
AGGRESSIVEVOICEDAILY_20050107.2012
AGGRESSIVEVOICEDAILY_20050109.1627
AGGRESSIVEVOICEDAILY_20050113.1400
AGGRESSIVEVOICEDAILY_20050114.1922
AGGRESSIVEVOICEDAILY_20050116.2149
AGGRESSIVEVOICEDAILY_20050124.1354
AGGRESSIVEVOICEDAILY_20050125.0136
AGGRESSIVEVOICEDAILY_20050203.1356
AGGRESSIVEVOICEDAILY_20050205.1954
AGGRESSIVEVOICEDAILY_20050208.1142
AGGRESSIVEVOICEDAILY_20050213.2123
AGGRESSIVEVOICEDAILY_20050224.1207
AGGRESSIVEVOICEDAILY_20050224.2252
BACONSREBELLION_20050123.1639
BACONSREBELLION_20050125.1108
BACONSREBELLION_20050127.1017
BACONSREBELLION_20050204.1326
BACONSREBELLION_20050205.1919
BACONSREBELLION_20050206.1345
BACONSREBELLION_20050209.0721
BACONSREBELLION_20050210.0728
BACONSREBELLION_20050214.0944
BACONSREBELLION_20050216.1536
BACONSREBELLION_20050216.1618
BACONSREBELLION_20050216.1632
BACONSREBELLION_20050217.0744
BACONSREBELLION_20050218.0848
BACONSREBELLION_20050218.1214
BACONSREBELLION_20050222.1348
BACONSREBELLION_20050227.1238
BACONSREBELLION_20050222.0817
BACONSREBELLION_20050226.1317
FLOPPINGACES_20041113.1528.042
FLOPPINGACES_20041114.1240.039
FLOPPINGACES_20041115.1613.032
FLOPPINGACES_20041116.0833.027

fsh_29171
fsh_29187
fsh_29191
fsh_29192
fsh_29195
fsh_29226
fsh_29272
fsh_29302
fsh_29303
fsh_29326
fsh_29336
fsh_29344
fsh_29348
fsh_29350
fsh_29361
fsh_29388
fsh_29395
fsh_29505
fsh_29520
fsh_29521
fsh_29526
fsh_29581_1
fsh_29586
fsh_29592
fsh_29601
fsh_29622
fsh_29628
fsh_29630
fsh_29770
fsh_29774
fsh_29782_2
fsh_29783
fsh_29786

bc

CNN_CF_20030303.1900.00
CNN_CF_20030303.1900.05
CNN_CF_20030303.1900.06-1
CNN_CF_20030303.1900.06-2
CNN_CF_20030304.1900.02
CNN_CF_20030304.1900.04
CNN_CF_20030304.1900.06-2
CNN_CF_20030305.1900.00-1
CNN_CF_20030305.1900.00-2
CNN_CF_20030305.1900.00-3
CNN_CF_20030305.1900.02
CNN_CF_20030305.1900.06-1
CNN_CF_20030305.1900.06-2
CNN_IP_20030328.1600.07
CNN_IP_20030329.1600.00-3
CNN_IP_20030329.1600.00-4
CNN_IP_20030329.1600.00-5
CNN_IP_20030329.1600.00-6
CNN_IP_20030329.1600.01-1
CNN_IP_20030329.1600.01-3
CNN_IP_20030329.1600.02
CNN_IP_20030330.1600.05-2
CNN_IP_20030330.1600.06
CNN_IP_20030402.1600.00-2
CNN_IP_20030402.1600.00-3
CNN_IP_20030402.1600.00-4
CNN_IP_20030402.1600.02-1
CNN_ENG_20030509_090025.5
CNN_ENG_20030509_123601.13
CNN_ENG_20030512_190454.7
CNN_ENG_20030513_080020.2
CNN_ENG_20030513_113501.6
CNN_ENG_20030513_160506.16
CNN_ENG_20030514_130518.5
CNN_ENG_20030515_063019.6
CNN_ENG_20030515_073019.7
CNN_ENG_20030515_193533.6
CNN_ENG_20030516_090022.7
CNN_ENG_20030516_123543.8
CNN_ENG_20030524_143511.4
CNN_ENG_20030525_143522.8
CNN_ENG_20030525_160525.13
CNN_ENG_20030526_133535.4
CNN_ENG_20030526_180540.6
CNN_ENG_20030526_183538.3
CNN_ENG_20030527_195948.3
CNN_ENG_20030527_215946.12
CNN_ENG_20030528_082823.9
CNN_ENG_20030528_125956.8
CNN_ENG_20030528_165958.16
CNN_ENG_20030528_172957.18
CNN_ENG_20030528_195959.20
CNN_ENG_20030529_085826.10
CNN_ENG_20030529_130011.6
CNN_ENG_20030530_130025.12
CNN_ENG_20030602_072826.1
CNN_ENG_20030602_102826.13
CNN_ENG_20030602_105829.2
CNN_ENG_20030602_133012.9
CNN_ENG_20030603_095830.17
CNN_ENG_20030603_133025.7
CNN_ENG_20030604_092828.7
CNN_ENG_20030604_102828.6
CNN_ENG_20030605_065831.18
CNN_ENG_20030605_085831.13
CNN_ENG_20030605_105831.11
CNN_ENG_20030605_193002.8
CNN_ENG_20030605_223004.4
CNN_ENG_20030607_170312.6
CNN_ENG_20030607_173310.4
CNN_ENG_20030610_085833.10
CNN_ENG_20030610_095857.4
CNN_ENG_20030610_105832.1
CNN_ENG_20030610_123040.9
CNN_ENG_20030610_130042.17
CNN_ENG_20030610_133041.17
CNN_ENG_20030611_102832.3
CNN_ENG_20030611_102832.4
CNN_ENG_20030611_175950.5
CNN_ENG_20030612_072835.2
CNN_ENG_20030612_160005.13
CNN_ENG_20030612_173004.10
CNN_ENG_20030612_173004.2
CNN_ENG_20030614_173123.4
CNN_ENG_20030616_130059.25
CNN_ENG_20030617_065838.21
CNN_ENG_20030617_105836.4
CNN_ENG_20030617_112838.4
CNN_ENG_20030617_173115.14
FLOPPINGACES_20041228.0927.010
FLOPPINGACES_20041230.1844.003
FLOPPINGACES_20050101.2244.048
GETTINGPOLITICAL_20050105.0127.001
OIADVANTAGE_20041224.1007
OIADVANTAGE_20050103.0944
OIADVANTAGE_20050105.0922
OIADVANTAGE_20050108.1323
OIADVANTAGE_20050109.1947
OIADVANTAGE_20050110.1009
OIADVANTAGE_20050203.1000
OIADVANTAGE_20050203.2102
OIADVANTAGE_20050204.1155
HEALINGIRAQ_20041108.1942.05
MARKBACKER_20041103.1300
MARKBACKER_20041108.1507
MARKBACKER_20041112.0707
MARKBACKER_20041117.0723
MARKBACKER_20041117.1107
MARKBACKER_20041119.1002
MARKBACKER_20041128.1641
MARKBACKER_20041202.0711
MARKBACKER_20041206.0733
MARKBACKER_20041216.0656
MARKBACKER_20041217.1639
MARKBACKER_20041220.0919
MARKBACKER_20050103.0829
MARKBACKER_20050105.1526
MARKBACKER_20050105.1632
MARKBACKER_20050217.0647
MARKETVIEW_20041209.1401
MARKETVIEW_20041211.1845
MARKETVIEW_20041212.1447
MARKETVIEW_20041213.0722
MARKETVIEW_20041215.2128
MARKETVIEW_20041217.0801
MARKETVIEW_20041219.1509
MARKETVIEW_20041220.1537
MARKETVIEW_20050105.1901
MARKETVIEW_20050120.1641
MARKETVIEW_20050126.0711
MARKETVIEW_20050127.0716
MARKETVIEW_20050201.0748
MARKETVIEW_20050204.1322
MARKETVIEW_20050204.1337
MARKETVIEW_20050204.1736
MARKETVIEW_20050205.1358
MARKETVIEW_20050206.1951
MARKETVIEW_20050206.2009
MARKETVIEW_20050207.0746
MARKETVIEW_20050208.2033
MARKETVIEW_20050208.2059
MARKETVIEW_20050209.1923
MARKETVIEW_20050210.2138
MARKETVIEW_20050212.1607
MARKETVIEW_20050212.1717
MARKETVIEW_20050214.2115
MARKETVIEW_20050215.1858
MARKETVIEW_20050216.2120
MARKETVIEW_20050217.2115
MARKETVIEW_20050222.0729
MARKETVIEW_20050222.1919

CNN_IP_20030402.1600.02-2
CNN_IP_20030403.1600.00-1
CNN_IP_20030403.1600.00-2
CNN_IP_20030403.1600.00-3
CNN_IP_20030403.1600.00-4
CNN_IP_20030404.1600.00-1
CNN_IP_20030404.1600.00-2
CNN_IP_20030405.1600.00-2
CNN_IP_20030405.1600.00-3
CNN_IP_20030405.1600.01-2
CNN_IP_20030405.1600.01-3
CNN_IP_20030405.1600.02
CNN_IP_20030406.1600.03
CNN_IP_20030407.1600.05
CNN_IP_20030408.1600.03
CNN_IP_20030408.1600.04
CNN_IP_20030409.1600.04
CNN_IP_20030410.1600.03-1
CNN_IP_20030410.1600.03-2
CNN_IP_20030412.1600.03
CNN_IP_20030412.1600.05
CNN_IP_20030414.1600.04
CNN_IP_20030417.1600.06
CNN_IP_20030422.1600.05
CNN_LE_20030504.1200.01
CNN_LE_20030504.1200.02-1
CNN_LE_20030504.1200.02-2
CNN_CF_20030304.1900.01

bn

CNN_ENG_20030305_170125.1
CNN_ENG_20030306_070606.18
CNN_ENG_20030306_083604.6
CNN_ENG_20030312_083725.3
CNN_ENG_20030312_223733.14
CNN_ENG_20030313_083739.0
CNN_ENG_20030318_140851.8
CNN_ENG_20030320_153434.7
CNN_ENG_20030325_150531.10
CNN_ENG_20030325_220534.6
CNN_ENG_20030327_163556.20
CNN_ENG_20030329_170349.7
CNN_ENG_20030331_123648.4
CNN_ENG_20030331_193655.14
CNN_ENG_20030401_073033.14
CNN_ENG_20030401_233449.5
CNN_ENG_20030402_190500.11
CNN_ENG_20030403_060032.0
CNN_ENG_20030403_080032.9
CNN_ENG_20030403_090032.1
CNN_ENG_20030403_180511.16
CNN_ENG_20030617_173115.22
CNN_ENG_20030617_193116.10
CNN_ENG_20030618_065839.11
CNN_ENG_20030618_150128.5
CNN_ENG_20030618_150128.6
CNN_ENG_20030618_193127.17
CNN_ENG_20030619_115954.10
CNN_ENG_20030619_115954.4
CNN_ENG_20030619_125955.10
CNN_ENG_20030620_095840.4
CNN_ENG_20030620_170011.14
CNN_ENG_20030621_115841.16
CNN_ENG_20030621_160254.25
CNN_ENG_20030622_173306.9
CNN_ENG_20030624_065843.24
CNN_ENG_20030624_082841.12
CNN_ENG_20030624_140104.22
CNN_ENG_20030624_153103.16
CNN_ENG_20030624_153103.17
CNN_ENG_20030625_210122.0
CNN_ENG_20030625_220123.3
CNN_ENG_20030626_193133.8
CNN_ENG_20030627_065846.3
CNN_ENG_20030627_130145.6
CNN_ENG_20030630_075848.7
CNN_ENG_20030630_085848.18
CNN_ENG_20030626_203133.11
CNN_ENG_20030605_153000.9
CNN_ENG_20030411_070039.21
CNNHL_ENG_20030312_150218.13
CNNHL_ENG_20030331_193419.9
CNNHL_ENG_20030402_133449.22
CNNHL_ENG_20030402_193443.5
CNNHL_ENG_20030403_133453.21
CNNHL_ENG_20030403_193455.30
CNNHL_ENG_20030407_193547.5
CNNHL_ENG_20030411_230640.38
CNNHL_ENG_20030415_193729.5
CNNHL_ENG_20030416_133739.13
CNNHL_ENG_20030416_133739.9
CNNHL_ENG_20030416_193742.26
CNNHL_ENG_20030416_193742.7
CNNHL_ENG_20030416_230741.33
CNNHL_ENG_20030425_183518.12
CNNHL_ENG_20030428_123600.14
CNNHL_ENG_20030429_220618.15
CNNHL_ENG_20030430_220712.37
CNNHL_ENG_20030505_220734.25
CNNHL_ENG_20030513_183907.5
CNNHL_ENG_20030513_220910.11
CNNHL_ENG_20030519_124020.23
MARKETVIEW_20050225.0541
MARKETVIEW_20050226.1307
MARKETVIEW_20050226.1444
MARKETVIEW_20050228.2211

un

Austin-Grad-Community_20050212.2454
Integritas-Group-Community-Forum_20050110.0557
alt.atheism_20041104.2428
alt.books.tom-clancy_20050130.1848
alt.collecting.autographs_20050224.2438
alt.corel_20041228.0503
alt.gossip.celebrities_20041118.2331
alt.gossip.celebrities_20050218.0826
alt.obituaries_20041121.1339
alt.politics.economics_20041206.1835
alt.politics_20050124.0640
alt.religion.mormon_20050103.0854
alt.support.divorce_20050113.2451
alt.sys.pc-clone.dell_20050226.2350
alt.vacation.las-vegas_20050109.0133
aus.cars_20041206.0903
misc.invest.marketplace_20050208.2406
misc.kids.pregnancy_20050120.0404
misc.legal.moderated_20041202.1648
misc.legal.moderated_20050129.2225
misc.survivalism_20050210.0232
misc.taxes_20050218.1250
rec.arts.mystery_20050219.1126
rec.arts.sf.written.robert-jordan_20050208.1350
rec.boats_20050130.1006
rec.music.makers.guitar.acoustic_20041228.1628
rec.music.phish_20041215.1554
rec.music.phish_20050217.1804
rec.parks.theme_20050217.2019
rec.sport.disc_20050209.2202
rec.travel.cruises_20050216.1636
rec.travel.cruises_20050222.0313
rec.travel.europe_20050101.1800
rec.travel.usa-canada_20050128.0121
seattle.politics_20050122.2412
soc.culture.china_20050203.0639
soc.culture.hmong_20050210.1130
soc.culture.indian_20041104.2348
soc.culture.iraq_20050211.0445
soc.culture.jewish_20050130.2105
soc.history.war.world-war-ii_20050127.2403
soc.history.what-if_20050129.1404
talk.politics.misc_20050216.1337
uk.gay-lesbian-bi_20050127.0311
marcellapr_20050211.2013

TABLE A.1: Data splits of ACE 2005 as used in [Ferguson et al., 2018; Nguyen et al., 2016; Feng et al., 2016] and other related works in event extraction. The file names refer to the versions located in the "timex2norm" corpus sub-directories.

Table A.2: Event Distribution in the ACE 2005 Data Splits

Event Type	ACE.TRAIN	ACE.TEST	ACE.DEV
Business.Declare-Bankruptcy	40	2	1
Business.End-Org	31	5	1
Business.Merge-Org	14	**0**	**0**
Business.Start-Org	29	18	**0**
Conflict.Attack	1273	93	177
Conflict.Demonstrate	65	7	9
Contact.Meet	200	50	30
Contact.Phone-Write	112	8	3
Justice.Acquit	5	1	**0**
Justice.Appeal	30	6	7
Justice.Arrest-Jail	78	6	4
Justice.Charge-Indict	96	8	2
Justice.Convict	64	6	6
Justice.Execute	14	2	5
Justice.Extradite	6	1	**0**
Justice.Fine	22	6	**0**
Justice.Pardon	2	0	**0**
Justice.Release-Parole	46	1	**0**
Justice.Sentence	84	11	4
Justice.Sue	60	4	12
Justice.Trial-Hearing	103	5	1
Life.Be-Born	47	3	**0**
Life.Die	524	17	57
Life.Divorce	20	9	**0**
Life.Injure	127	1	14
Life.Marry	73	10	**0**
Movement.Transport	611	48	62
Personnel.Elect	162	16	5
Personnel.End-Position	159	22	31
Personnel.Nominate	11	1	**0**
Personnel.Start-Position	92	13	13
Transaction.Transfer-Money	128	14	56
Transaction.Transfer-Ownership	92	30	5

TABLE A.2: Event distribution in the ACE 2005 train, test, and development set splits used in [Ferguson et al., 2018; Nguyen et al., 2016; Feng et al., 2016] and other related works in event extraction. The development set does not contain any mentions for 11 out of the 33 ACE event types.

List of Figures

List of Tables

List of Abbreviations

ANN Artificial Neural Network

Bi-LSTM Bi-directional Long Short-Term Memory network

Bi-RNN Bi-directional Recurrent Neural Network

BPTT Back-Propagation Through Time

CNN Convolutional Neural Network

CR Coreference Resolution

FFNN Feed-forward Neural Network

IE Information Extraction

LSTM Long Short-Term Memory network

MLP Multi-Layer Perceptron

MWE Multi-Word Expression

NE Named Entity

NER Named Entity Recognition

NLP Natural Language Processing

PCA Principal Component Analysis

POS Part Of Speech

RE Relation Extraction

RNN Recurrent Neural Network

Bibliography

Abney, S. (2007). Semisupervised Learning for Computational Linguistics. 1st edition, Chapman & Hall, CRC.

Agichtein, E. and Gravano, L. (2000). Snowball: Extracting Relations from Large Plain-Text Collections. In Proceedings of the Fifth ACM Conference on Digital Libraries DL '00 pp. 85–94, Association for Computing Machinery, New York, NY, USA.

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M. and Soroa, A. (2009). A Study on Similarity and Relatedness Using Distributional and WordNet-Based Approaches. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics NAACL '09 pp. 19–27, Association for Computational Linguistics, USA.

Agirre, E. and Martinez, D. (2001). Learning class-to-class selectional preferences. In Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL) Association for Computational Linguistics.

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S. and Vollgraf, R. (2019). FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations) pp. 54–59, Association for Computational Linguistics, Minneapolis, Minnesota.

Akbik, A., Blythe, D. and Vollgraf, R. (2018). Contextual String Embeddings for Sequence Labeling. In Proceedings of the 27th International Conference on Computational Linguistics pp. 1638–1649, Association for Computational Linguistics, Santa Fe, New Mexico, USA.

Almor, A. (1999). Noun-Phrase Anaphora and Focus: The Informational Load Hypothesis. Psychological Review *106*.

Aone, C. and William, S. (1995). Evaluating Automated and Manual Acquisition of Anaphora Resolution Strategies. In 33rd Annual Meeting of the Association for Computational Linguistics pp. 122–129, Association for Computational Linguistics, Cambridge, Massachusetts, USA.

Ariel, M. (1990). Accessing Noun-Phrase Antecedents. Routledge.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. In Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference ISWC'07/ASWC'07 pp. 722–735, Springer-Verlag, Berlin, Heidelberg.

Bagga, A. and Baldwin, B. (1998). Algorithms for Scoring Coreference Chains. In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference pp. 563–566, European Language Resources Association (ELRA).

Baker, C. F., Fillmore, C. J. and Lowe, J. B. (1998). The Berkeley FrameNet Project. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1 ACL '98/COLING '98 pp. 86–90, Association for Computational Linguistics, USA.

Bansal, M. and Klein, D. (2012). Coreference Semantics from Web Features. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 389–398, Association for Computational Linguistics, Jeju Island, Korea.

Baroni, M., Bernardini, S., Ferraresi, A. and Zanchetta, E. (2009). The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. Language Resources and Evaluation *43*, 209–226.

Baroni, M., Dinu, G. and Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 238–247, Association for Computational Linguistics, Baltimore, Maryland.

Bean, D. and Riloff, E. (2004). Unsupervised Learning of Contextual Role Knowledge for Coreference Resolution. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004 pp. 297–304, Association for Computational Linguistics, Boston, Massachusetts, USA.

Bejan, C. and Harabagiu, S. (2010). Unsupervised Event Coreference Resolution with Rich Linguistic Features. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics pp. 1412–1422, Association for Computational Linguistics, Uppsala, Sweden.

Bengio, Y., Ducharme, R., Vincent, P. and Janvin, C. (2003). A Neural Probabilistic Language Model. J. Mach. Learn. Res. *3*, 1137–1155.

Bengtson, E. and Roth, D. (2008). Understanding the Value of Features for Coreference Resolution. In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing pp. 294–303, Association for Computational Linguistics, Honolulu, Hawaii.

Berant, J., Srikumar, V., Chen, P.-C., Vander Linden, A., Harding, B., Huang, B., Clark, P. and Manning, C. D. (2014). Modeling Biological Processes for Reading Comprehension. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) pp. 1499–1510, Association for Computational Linguistics, Doha, Qatar.

Bergsma, S. and Lin, D. (2006). Bootstrapping Path-Based Pronoun Resolution. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics ACL-44 pp. 33–40, Association for Computational Linguistics, USA.

Bergsma, S., Lin, D. and Goebel, R. (2008). Discriminative Learning of Selectional Preference from Unlabeled Text. In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing pp. 59–68, Association for Computational Linguistics, Honolulu, Hawaii.

Björkelund, A. and Farkas, R. (2012). Data-driven Multilingual Coreference Resolution using Resolver Stacking. In Joint Conference on EMNLP and CoNLL - Shared Task pp. 49–55, Association for Computational Linguistics, Jeju Island, Korea.

Björkelund, A. and Kuhn, J. (2014). Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) Association for Computational Linguistics.

Blum, A. and Mitchell, T. (1998). Combining Labeled and Unlabeled Data with Co-training. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory COLT' 98 pp. 92–100, ACM, New York, NY, USA.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics *5*, 135–146.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T. and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In SIGMOD Conference pp. 1247–1250, ACM.

Brants, T. and Franz., A. (2006). The Google Web 1T 5-gram corpus version 1.1. Technical report Philadelphia: Linguistic Data Consortium.

Brin, S. (1998). Extracting Patterns and Relations from the World Wide Web. In WebDB pp. 172–183, ACM.

Brown, P. F., Della Pietra, V. J., deSouza, P. V., Lai, J. C. and Mercer, R. L. (1992). Class-Based n-gram Models of Natural Language. Computational Linguistics *18*, 467–480.

Bryl, V., Giuliano, C., Serafini, L. and Tymoshenko, K. (2010a). Supporting Natural Language Processing with Background Knowledge: Coreference Resolution Case. In Proceedings of the 9th International Semantic Web Conference (ISWC2010) pp. 80–95, Springer-Verlag.

Bryl, V., Giuliano, C., Serafini, L. and Tymoshenko, K. (2010b). Using Background Knowledge to Support Coreference Resolution. In ECAI 2010 vol. 215, pp. 759–764, IOS Press.

Camacho-Collados, J. and Pilehvar, M. T. (2018). From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. J. Artif. Int. Res. *63*, 743–788.

Casati, R. and Varzi, A. (2015). Events. In The Stanford Encyclopedia of Philosophy, (Zalta, E. N., ed.),. Metaphysics Research Lab, Stanford University winter 2015 edition.

Chen, C., Zhang, Y. and Gao, Y. (2018). Learning How to Self-Learn: Enhancing Self-Training Using Neural Reinforcement Learning. In International Conference on Asian Language Processing (IALP) pp. 25–30, IEEE.

Chen, Y., Liu, S., Zhang, X., Liu, K. and Zhao, J. (2017). Automatically Labeled Data Generation for Large Scale Event Extraction. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 409–419, Association for Computational Linguistics, Vancouver, Canada.

Chen, Y., Xu, L., Liu, K., Zeng, D. and Zhao, J. (2015). Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) pp. 167–176, Association for Computational Linguistics, Beijing, China.

Chinchor, N. (2001). Message Understanding Conference (MUC) 7 LDC2001T02. Technical report Philadelphia: Linguistic Data Consortium.

Chinchor, N. and Sundheim, B. (2003). Message Understanding Conference (MUC) 6 LDC2003T13. Technical report Philadelphia: Linguistic Data Consortium.

Clark, K. and Manning, C. D. (2016a). Deep Reinforcement Learning for Mention-Ranking Coreference Models. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing pp. 2256–2262, Association for Computational Linguistics, Austin, Texas.

Clark, K. and Manning, C. D. (2016b). Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 643–653, Association for Computational Linguistics, Berlin, Germany.

Clark, S., Curran, J. and Osborne, M. (2003). Bootstrapping POS-taggers using unlabelled data. In Proceedings of the Seventh Conference on Natural Language

Learning at HLT-NAACL 2003 pp. 49–55, Association for Computational Linguistics.

Collobert, R. and Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In Proceedings of the 25th International Conference on Machine Learning ICML '08 pp. 160–167, Association for Computing Machinery, New York, NY, USA.

Culotta, A., Wick, M. and McCallum, A. (2007). First-Order Probabilistic Models for Coreference Resolution. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference pp. 81–88, Association for Computational Linguistics, Rochester, New York.

Cybulska, A. and Vossen, P. (2014). 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'14) pp. 4545–4552, European Language Resources Association (ELRA), Reykjavik, Iceland.

Dagan, I. and Itai, A. (1990). Automatic Processing of Large Corpora for the Resolution of Anaphora References. In COLING 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics Association for Computational Linguistics.

Daumé III, H. and Marcu, D. (2005). A Large-scale Exploration of Effective Global Features for a Joint Entity Detection and Tracking Model. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing HLT '05 Association for Computational Linguistics, Stroudsburg, PA, USA.

Denis, P. and Baldridge, J. (2007). A Ranking Approach to Pronoun Resolution. In IJCAI Morgan Kaufmann Publishers Inc.

Denis, P. and Baldridge, J. (2008). Specialized Models and Ranking for Coreference Resolution. In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing pp. 660–669, Association for Computational Linguistics, Honolulu, Hawaii.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings

of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) pp. 4171–4186, Association for Computational Linguistics, Minneapolis, Minnesota.

Dhillon, I. S. and Modha, D. S. (2001). Concept Decompositions for Large Sparse Text Data Using Clustering. Mach. Learn. *42*, 143–175.

Dror, R., Baumer, G., Shlomov, S. and Reichart, R. (2018). The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 1383–1392, Association for Computational Linguistics, Melbourne, Australia.

Durrett, G., Hall, D. and Klein, D. (2013). Decentralized Entity-Level Modeling for Coreference Resolution. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 114–124, Association for Computational Linguistics, Sofia, Bulgaria.

Durrett, G. and Klein, D. (2013). Easy Victories and Uphill Battles in Coreference Resolution. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing pp. 1971–1982, Association for Computational Linguistics, Seattle, Washington, USA.

Durrett, G. and Klein, D. (2014). A Joint Model for Entity Analysis: Coreference, Typing, and Linking. Transactions of the Association for Computational Linguistics *2*, 477–490.

Erk, K. (2012). Vector Space Models of Word Meaning and Phrase Meaning: A Survey. Lang. Linguistics Compass *6*, 635–653.

Fei, H., Li, X., Li, D. and Li, P. (2019). End-to-end Deep Reinforcement Learning Based Coreference Resolution. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics pp. 660–665, Association for Computational Linguistics, Florence, Italy.

Feng, X., Huang, L., Tang, D., Ji, H., Qin, B. and Liu, T. (2016). A Language-Independent Neural Network for Event Detection. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) pp. 66–71, Association for Computational Linguistics, Berlin, Germany.

Ferguson, J., Lockard, C., Hawkins, N., Soderland, S., Hajishirzi, H. and Weld, D. S. (2016). University of Washington TAC-KBP 2016 System Description. In Proceedings of the 2016 Text Analysis Conference, TAC 2016, Gaithersburg, Maryland, USA, November 14-15, 2016 NIST.

Ferguson, J., Lockard, C., Weld, D. and Hajishirzi, H. (2018). Semi-Supervised Event Extraction with Paraphrase Clusters. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers) pp. 359–364, Association for Computational Linguistics, New Orleans, Louisiana.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G. and Ruppin, E. (2001). Placing Search in Context: The Concept Revisited. In Proceedings of the 10th International Conference on World Wide Web WWW '01 pp. 406–414, Association for Computing Machinery.

Finley, T. and Joachims, T. (2005). Supervised Clustering with Support Vector Machines. In Proceedings of the 22nd International Conference on Machine Learning ICML '05 pp. 217–224, Association for Computing Machinery, New York, NY, USA.

Fleischman, M., Hovy, E. and Echihabi, A. (2003). Offline Strategies for Online Question Answering: Answering Questions Before They Are Asked. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics pp. 1–7, Association for Computational Linguistics, Sapporo, Japan.

Fujiwara, S. and Sekine, S. (2011). Self-adjusting Bootstrapping. In Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part II, (Gelbukh, A. F., ed.), vol. 6609, of Lecture Notes in Computer Science pp. 188–201, Springer.

Garera, N. and Yarowsky, D. (2006). Resolving and Generating Definite Anaphora by Modeling Hypernymy using Unlabeled Corpora. In Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X) pp. 37–44, Association for Computational Linguistics, New York City.

Ghoulam, A., Barigou, F. and Belalem, G. (2015). Information Extraction in the Medical Domain. Journal of Information Technology Research *8*, 1–15.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org.

Graves, A. (2012). Supervised Sequence Labelling with Recurrent Neural Networks. Springer Berlin Heidelberg, Berlin, Heidelberg.

Grice, H. P. (1975). Logic and Conversation. In Syntax and Semantics: Vol. 3: Speech Acts, (Cole, P. and Morgan, J. L., eds), pp. 41–58. Academic Press New York.

Guo, J., Che, W., Wang, H. and Liu, T. (2014). Revisiting Embedding Features for Simple Semi-supervised Learning. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) pp. 110–120, Association for Computational Linguistics, Doha, Qatar.

Gupta, A., Dengre, V., Kheruwala, H. A. and Shah, M. (2020). Comprehensive review of text-mining applications in finance. Financial Innovation *6*.

Hacker, P. M. S. (1982). Events, Ontology and Grammar. Philosophy *57*, 477–486.

Haghighi, A. and Klein, D. (2009). Simple Coreference Resolution with Rich Syntactic and Semantic Features. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing pp. 1152–1161, Association for Computational Linguistics, Singapore.

Halliday, M. and Hasan, R. (1976). Cohesion in English. English Language Series: A Longman Paperback, Longman.

Harabagiu, S. M., Bunescu, R. C. and Maiorano, S. J. (2001). Text and Knowledge Mining for Coreference Resolution. In Second Meeting of the North American Chapter of the Association for Computational Linguistics Association for Computational Linguistics.

Harris, Z. S. (1954). Distributional Structure. Word *10*, 146–162.

Hearst, M. A. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora. In COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics Association for Computational Linguistics.

Hedderich, M. A. and Klakow, D. (2018). Training a Neural Network in a Low-Resource Setting on Automatically Annotated Noisy Data. In Proceedings of

the Workshop on Deep Learning Approaches for Low-Resource NLP pp. 12–18, Association for Computational Linguistics, Melbourne.

Heinzerling, B., Moosavi, N. S. and Strube, M. (2017). Revisiting Selectional Preferences for Coreference Resolution. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing pp. 1332–1339, Association for Computational Linguistics, Copenhagen, Denmark.

Hobbs, J. (1986). Resolving Pronoun References pp. 339–352. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Comput., *9*, 1735–1780.

Holen, G. I. (2013). Critical Reflections on Evaluation Practices in Coreference Resolution. In Proceedings of the 2013 NAACL HLT Student Research Workshop, pp. 1–7, Association for Computational Linguistics, Atlanta, Georgia.

Hong, Y., Zhang, J., Ma, B., Yao, J., Zhou, G. and Zhu, Q. (2011). Using Cross-Entity Inference to Improve Event Extraction. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies pp. 1127–1136, Association for Computational Linguistics, Portland, Oregon, USA.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L. and Weischedel, R. (2006). OntoNotes: The 90% Solution. In Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers NAACL-Short '06 pp. 57–60, Association for Computational Linguistics, USA.

Iida, R., Inui, K. and Matsumoto, Y. (2009). Capturing Salience with a Trainable Cache Model for Zero-anaphora Resolution. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP pp. 647–655, Association for Computational Linguistics, Suntec, Singapore.

Jespersen, O. (1949). A Modern Grammar On Historical Principles. Part 7 Syntax. Taylor and Francis Ltd, London, UK.

Ji, H. and Grishman, R. (2008). Refining Event Extraction through Cross-Document Inference. In Proceedings of ACL-08: HLT pp. 254–262, Association for Computational Linguistics, Columbus, Ohio.

Joshi, M., Levy, O., Zettlemoyer, L. and Weld, D. (2019). BERT for Coreference Resolution: Baselines and Analysis. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) pp. 5803–5808, Association for Computational Linguistics, Hong Kong, China.

Kantor, B. and Globerson, A. (2019). Coreference Resolution with Entity Equalization. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics pp. 673–677, Association for Computational Linguistics, Florence, Italy.

Krause, S. (2018). Knowledge-Intensive, High-Performance Relation Extraction. PhD thesis, Technischen Universität Berlin Berlin.

Krause, S., Li, H., Uszkoreit, H. and Xu, F. (2012a). Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web. In Proceedings of the 11th International Semantic Web Conference Springer.

Krause, S., Li, H., Uszkoreit, H. and Xu, F. (2012b). Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web. In The Semantic Web – ISWC 2012, (Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J. X., Hendler, J., Schreiber, G., Bernstein, A. and Blomqvist, E., eds), pp. 263–278, Springer Berlin Heidelberg, Berlin, Heidelberg.

Krause, S., Li, H., Xu, F., Uszkoreit, H., Hummel, R. and Spielhagen, L. (2014). Language Resources and Annotation Tools for Cross-Sentence Relation Extraction. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), (Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J. and Piperidis, S., eds), European Language Resources Association (ELRA), Reykjavik, Iceland.

Krause, S., Xu, F., Uszkoreit, H. and Weissenborn, D. (2016). Event Linking with Sentential Features from Convolutional Neural Networks. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning pp. 239–249, Association for Computational Linguistics, Berlin, Germany.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C. (2016). Neural Architectures for Named Entity Recognition. In Proceedings of the 2016

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 260–270, Association for Computational Linguistics, San Diego, California.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R. (2020). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In International Conference on Learning Representations ICLR.

Landauer, T. K. and Dumais, S. T. (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. Psychological Review *104*, 211–240.

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M. and Jurafsky, D. (2011). Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task. In Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task pp. 28–34, Association for Computational Linguistics, Portland, Oregon, USA.

Lee, K., He, L., Lewis, M. and Zettlemoyer, L. (2017). End-to-end Neural Coreference Resolution. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing pp. 188–197, Association for Computational Linguistics, Copenhagen, Denmark.

Lee, K., He, L. and Zettlemoyer, L. (2018). Higher-Order Coreference Resolution with Coarse-to-Fine Inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers) pp. 687–692, Association for Computational Linguistics, New Orleans, Louisiana.

Levy, O. and Goldberg, Y. (2014). Dependency-Based Word Embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) pp. 302–308, Association for Computational Linguistics, Baltimore, Maryland.

Li, Q., Ji, H. and Huang, L. (2013). Joint Event Extraction via Structured Prediction with Global Features. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 73–82, Association for Computational Linguistics, Sofia, Bulgaria.

Liao, S. and Grishman, R. (2010). Using Document Level Cross-Event Inference to Improve Event Extraction. In Proceedings of the 48th Annual Meeting of the

Association for Computational Linguistics pp. 789–797, Association for Computational Linguistics, Uppsala, Sweden.

Liao, S. and Grishman, R. (2011). Can Document Selection Help Semi-supervised Learning? A Case Study On Event Extraction. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies pp. 260–265, Association for Computational Linguistics, Portland, Oregon, USA.

Liu, P., Qiu, X. and Huang, X. (2015). Learning Context-Sensitive Word Embeddings with Neural Tensor Skip-Gram Model. In Proceedings of the 24th International Conference on Artificial Intelligence IJCAI'15 pp. 1284–1290, AAAI Press.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv *abs/1907.11692*.

Luo, X. (2005). On Coreference Resolution Performance Metrics. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing HLT '05 pp. 25–32, Association for Computational Linguistics, USA.

Luo, X., Ittycheriah, A., Jing, H., Kambhatla, N. and Roukos, S. (2004). A Mention-Synchronous Coreference Resolution Algorithm Based On the Bell Tree. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04) pp. 135–142, Association for Computational Linguistics, Barcelona, Spain.

Ma, X. and Hovy, E. (2016). 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) pp. 1064–1074, Association for Computational Linguistics, Berlin, Germany.

Mani, I., Pustejovsky, J. and Gaizauskas, R. (2005). The Language of Time: A Reader. Oxford University Press.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In

Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations pp. 55–60, Association for Computational Linguistics, Baltimore, Maryland.

Markert, K. and Nissim, M. (2005). Comparing Knowledge Sources for Nominal Anaphora Resolution. Computational Linguistics *31*, 367–402.

Martschat, S., Claus, P. and Strube, M. (2015). Plug Latent Structures and Play Coreference Resolution. In Proceedings of ACL-IJCNLP 2015 System Demonstrations pp. 61–66, Association for Computational Linguistics and The Asian Federation of Natural Language Processing, Beijing, China.

Martschat, S. and Strube, M. (2014). Recall Error Analysis for Coreference Resolution. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) pp. 2070–2081, Association for Computational Linguistics, Doha, Qatar.

Martschat, S. and Strube, M. (2015). Latent Structures for Coreference Resolution. Transactions of the Association for Computational Linguistics *3*, 405–418.

McCallum, A. and Wellner, B. (2005). Conditional Models of Identity Uncertainty with Application to Noun Coreference. In Advances in Neural Information Processing Systems 17, (Saul, L. K., Weiss, Y. and Bottou, L., eds), pp. 905–912. MIT Press.

McCarthy, J. and Lehnert, W. (1995). Using decision trees for coreference resolution. In Proceedings of the Fourteenth International Conference on Artificial Intelligence (IJCAI) pp. 1050–1055, Morgan Kaufmann Publishers Inc.

McClosky, D., Charniak, E. and Johnson, M. (2006a). Effective Self-training for Parsing. In Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics HLT-NAACL '06 pp. 152–159, Association for Computational Linguistics, Stroudsburg, PA, USA.

McClosky, D., Charniak, E. and Johnson, M. (2006b). Reranking and Self-training for Parser Adaptation. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics ACL-44 pp. 337–344, Association for Computational Linguistics, Stroudsburg, PA, USA.

Melli, G., Ester, M. and Sarkar, A. (2007). Recognition of Multi-sentence n-ary Subcellular Localization Mentions in Biomedical Abstracts. In Short Paper Proceedings of the 2nd International Symposium on Languages in Biology and Medicine (LBM 2007), Singapore, December 6-7, 2007, (Baker, C. J. O. and Su, J., eds), vol. 319, of CEUR Workshop Proceedings CEUR-WS.org.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. In 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, (Bengio, Y. and LeCun, Y., eds), International Conference on Learning Representations.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013b). Distributed Representations of Words and Phrases and Their Compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems NIPS'13 Curran Associates Inc.

Miller, G. A. (1995). WordNet: A Lexical Database for English. Commun. ACM *38*, 39–41.

Miller, G. A. and Hristea, F. (2006). Squibs and Discussions: WordNet Nouns: Classes and Instances. Computational Linguistics *32*, 1–3.

Mintz, M., Bills, S., Snow, R. and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP pp. 1003–1011, Association for Computational Linguistics, Suntec, Singapore.

Mitamura, T., Liu, Z. and Hovy, E. H. (2015). Overview of TAC KBP 2015 Event Nugget Track. In Proceedings of Text Analysis Conference NIST.

Mitchell, A., Strassel, S., Przybocki, M., Davis, J., Doddington, G. R., Grishman, R., Meyers, A., Brunstein, A., Ferro, L. and Sundheim, B. (2004). TIDES Extraction (ACE) 2003 Multilingual Training Data LDC2004T09. Technical report Philadelphia: Linguistic Data Consortium.

Mitkov, R. (2002). Anaphora Resolution. Routledge.

Mnih, A. and Hinton, G. (2008). A Scalable Hierarchical Distributed Language Model. In Proceedings of the 21st International Conference on Neural Information Processing Systems NIPS'08 pp. 1081–1088, Curran Associates Inc., Red Hook, NY, USA.

Moosavi, N. S. and Strube, M. (2016). Which Coreference Evaluation Metric Do You Trust? A Proposal for a Link-based Entity Aware Metric. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 632–642, Association for Computational Linguistics, Berlin, Germany.

Nasar, Z., Jaffry, S. W. and Malik, M. (2018). Information extraction from scientific articles: a survey. Scientometrics *117*.

Ng, V. (2004). Learning Noun Phrase Anaphoricity to Improve Conference Resolution: Issues in Representation and Optimization. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04) pp. 151–158, Association for Computational Linguistics, Barcelona, Spain.

Ng, V. (2007). Shallow Semantics for Coreference Resolution. In Proceedings of the 20th International Joint Conference on Artifical Intelligence IJCAI'07 pp. 1689–1694, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ng, V. (2010). Supervised Noun Phrase Coreference Research: The First Fifteen Years. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics pp. 1396–1411, Association for Computational Linguistics, Uppsala, Sweden.

Ng, V. and Cardie, C. (2002a). Identifying Anaphoric and Non-Anaphoric Noun Phrases to Improve Coreference Resolution. In COLING 2002: The 19th International Conference on Computational Linguistics Association for Computational Linguistics.

Ng, V. and Cardie, C. (2002b). Improving Machine Learning Approaches to Coreference Resolution. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics pp. 104–111, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA.

Ng, V. and Cardie, C. (2003). Weakly Supervised Natural Language Learning Without Redundant Views. In Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics pp. 173–180, Association for Computational Linguistics.

Nguyen, T. H., Cho, K. and Grishman, R. (2016). Joint Event Extraction via Recurrent Neural Networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 300–309, Association for Computational Linguistics, San Diego, California.

Palmer, M., Gildea, D. and Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. Computational Linguistics *31*, 71–106.

Parker, R., Graff, D., Kong, J., Chen, K. and Maeda, K. (2011). English Gigaword Fifth Edition LDC2011T07. Technical report Philadelphia: Linguistic Data Consortium.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A. (2017). Automatic Differentiation in PyTorch. In NIPS Autodiff Workshop Curran Associates Inc.

Peng, N., Poon, H., Quirk, C., Toutanova, K. and Yih, W. (2017). Cross-Sentence N-ary Relation Extraction with Graph LSTMs. TACL *5*, 101–115.

Pennington, J., Socher, R. and Manning, C. (2014). Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) pp. 1532–1543, Association for Computational Linguistics, Doha, Qatar.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) pp. 2227–2237, Association for Computational Linguistics, New Orleans, Louisiana.

Plank, B. and Agić, Ž. (2018). Distant Supervision from Disparate Sources for Low-Resource Part-of-Speech Tagging. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing pp. 614–620, Association for Computational Linguistics, Brussels, Belgium.

Plank, B., Søgaard, A. and Goldberg, Y. (2016). Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) pp. 412–418, Association for Computational Linguistics, Berlin, Germany.

Ponzetto, S. P. and Strube, M. (2006). Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution. In Proceedings of the Human Language Technology Conference of the NAACL, Main Conference pp. 192–199, Association for Computational Linguistics, New York City, USA.

Pradhan, S., Luo, X., Recasens, M., Hovy, E., Ng, V. and Strube, M. (2014). Scoring Coreference Partitions of Predicted Mentions: A Reference Implementation. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) pp. 30–35, Association for Computational Linguistics, Baltimore, Maryland.

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O. and Zhang, Y. (2012). CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. In Joint Conference on EMNLP and CoNLL - Shared Task CoNLL'12 pp. 1–40, Association for Computational Linguistics, USA.

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R. and Xue, N. (2011). CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task CoNLL'11 pp. 1–27, Association for Computational Linguistics, USA.

Quirk, C. and Poon, H. (2017). Distant Supervision for Relation Extraction beyond the Sentence Boundary. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers pp. 1171–1182, Association for Computational Linguistics, Valencia, Spain.

Rahman, A. and Ng, V. (2009). Supervised Models for Coreference Resolution. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing pp. 968–977, Association for Computational Linguistics, Singapore.

Rahman, A. and Ng, V. (2011). Coreference Resolution with World Knowledge. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies pp. 814–824, Association for Computational Linguistics, Portland, Oregon, USA.

Ratinov, L. and Roth, D. (2012). Learning-based Multi-Sieve Co-reference Resolution with Knowledge. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning pp. 1234–1244, Association for Computational Linguistics, Jeju Island, Korea.

Recasens, M., Can, M. and Jurafsky, D. (2013). Same Referent, Different Words: Unsupervised Mining of Opaque Coreferent Mentions. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 897–906, Association for Computational Linguistics, Atlanta, Georgia.

Recasens, M. and Hovy, E. (2009). A Deeper Look into Features for Coreference Resolution. In DAARC Springer-Verlag.

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

Reichart, R. and Rappoport, A. (2007). Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics pp. 616–623, Association for Computational Linguistics, Prague, Czech Republic.

Resnik, P. (1993). Selection and Information: A Class-based Approach to Lexical Relationships. PhD thesis, Department of Computer and Information Science, University of Pennsylvania Philadelphia, Penn.

Riedel, S., Yao, L. and McCallum, A. (2010). Modeling Relations and Their Mentions without Labeled Text. In Machine Learning and Knowledge Discovery in Databases, (Balcázar, J. L., Bonchi, F., Gionis, A. and Sebag, M., eds), pp. 148–163, Springer Berlin Heidelberg, Berlin, Heidelberg.

Rosen, S. (1999). The Syntactic Representation of Linguistic Events. Glot International *4*.

Ruder, S. and Plank, B. (2018). Strong Baselines for Neural Semi-Supervised Learning under Domain Shift. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) pp. 1044–1054, Association for Computational Linguistics, Melbourne, Australia.

Samad Zadeh Kaljahi, R. (2010). Adapting Self-Training for Semantic Role Labeling. In Proceedings of the ACL 2010 Student Research Workshop pp. 91–96, Association for Computational Linguistics, Uppsala, Sweden.

Sandhaus, E. (2008). The New York Times Annotated Corpus LDC2008T19. Technical report Philadelphia: Linguistic Data Consortium.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv *abs/1910.01108*.

Schnabel, T., Labutov, I., Mimno, D. and Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing pp. 298–307, Association for Computational Linguistics, Lisbon, Portugal.

Schuler, K. K. and Palmer, M. S. (2005). Verbnet: A Broad-Coverage, Comprehensive Verb Lexicon. PhD thesis, University of Pennsylvania USA. AAI3179808.

Simova, I. and Uszkoreit, H. (2017). Word Embeddings as Features for Supervised Coreference Resolution. In Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017 pp. 686–693, INCOMA Ltd., Varna, Bulgaria.

Søgaard, A. (2010). Simple Semi-supervised Training of Part-of-speech Taggers. In Proceedings of the ACL 2010 Conference Short Papers ACLShort '10 pp. 205–208, Association for Computational Linguistics, Stroudsburg, PA, USA.

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

Srivastava, R. K., Greff, K. and Schmidhuber, J. (2015). Training Very Deep Networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 NIPS'15 pp. 2377–2385, MIT Press, Cambridge, MA, USA.

Stoyanov, V., Gilbert, N., Cardie, C. and Riloff, E. (2009). Conundrums in Noun Phrase Coreference Resolution: Making Sense of the State-of-the-Art. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2 ACL '09 pp. 656–664, Association for Computational Linguistics, USA.

Strube, M. and Ponzetto, S. P. (2006). WikiRelate! Computing Semantic Relatedness Using Wikipedia. In Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2 AAAI'06 pp. 1419–1424, AAAI Press.

Stylianou, N. and Vlahavas, I. (2019). A Neural Entity Coreference Resolution Review. ArXiv *abs/1910.09329*.

Suchanek, F. M., Kasneci, G. and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In Proceedings of the 16th International Conference on World Wide Web WWW '07 pp. 697–706, Association for Computing Machinery, New York, NY, USA.

Sudo, K., Sekine, S. and Grishman, R. (2001). Automatic Pattern Acquisition for Japanese Information Extraction. In Proceedings of the First International Conference on Human Language Technology Research Association for Computational Linguistics.

Sudo, K., Sekine, S. and Grishman, R. (2003). An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics Association for Computational Linguistics.

Šuster, S., Titov, I. and van Noord, G. (2016). Bilingual Learning of Multi-sense Embeddings with Discrete Autoencoders. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 1346–1356, Association for Computational Linguistics, San Diego, California.

Swampillai, K. and Stevenson, M. (2010). Inter-sentential Relations in Information Extraction Corpora. In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10) European Language Resources Association (ELRA), Valletta, Malta.

Swampillai, K. and Stevenson, M. (2011). Extracting Relations Within and Across Sentences. In Proceedings of the International Conference Recent Advances in Natural Language Processing 2011 pp. 25–32, Association for Computational Linguistics, Hissar, Bulgaria.

Turian, J., Ratinov, L. and Bengio, Y. (2010). Word Representations: A Simple and General Method for Semi-supervised Learning. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics ACL '10 pp. 384–394, Association for Computational Linguistics, Stroudsburg, PA, USA.

Turney, P. D. and Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. J. Artif. Int. Res. *37*, 141–188.

Uryupina, O. (2010). Corry: A System for Coreference Resolution. In Proceedings of the 5th International Workshop on Semantic Evaluation SemEval '10 pp. 100–103, Association for Computational Linguistics.

Uryupina, O., Poesio, M., Giuliano, C. and Tymoshenko, K. (2011). Disambiguation and Filtering Methods in Using Web Knowledge for Coreference Resolution. In Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference Association for the Advancement of Artificial Intelligence.

Van de Cruys, T., Poibeau, T. and Korhonen, A. (2011). Latent Vector Weighting for Word Meaning in Context. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing pp. 1012–1022, Association for Computational Linguistics, Edinburgh, Scotland, UK.

van Deemter, K. and Kibble, R. (2000). On Coreferring: Coreference in MUC and Related Annotation Schemes. Computational Linguistics *26*, 629–637.

van der Goot, R., Plank, B. and Nissim, M. (2017). To normalize, or not to normalize: The impact of normalization on Part-of-Speech tagging. In Proceedings of the 3rd Workshop on Noisy User-generated Text pp. 31–39, Association for Computational Linguistics, Copenhagen, Denmark.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u. and Polosukhin, I. (2017). Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems NIPS'17 pp. 6000–6010, Curran Associates Inc., Red Hook, NY, USA.

Vendler, Z. (1957). Verbs and Times. The Philosophical Review *66*, 143–160.

Versley Y., Poesio M., P. S. (2016). Using Lexical and Encyclopedic Knowledge. In Anaphora Resolution. Theory and Applications of Natural Language Processing. Springer.

Vieira, R. and Poesio, M. (2000). An Empirically-based System for Processing Definite Descriptions. Computational Linguistics *26*, 539–593.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D. and Hirschman, L. (1995). A model-theoretic coreference scoring scheme. In Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995 pp. 45–52, MUC.

Walker, C., Strassel, S., Medero, J. and Maeda, K. (2006). ACE 2005 Multilingual Training Corpus LDC2006T06. Technical report Philadelphia: Linguistic Data Consortium.

Wang, Z., Shang, J., Liu, L., Lu, L., Liu, J. and Han, J. (2019). CrossWeigh: Training Named Entity Tagger from Imperfect Annotations. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) pp. 5154–5163, Association for Computational Linguistics, Hong Kong, China.

Weischedel, R., Hovy, E., Marcus, M., Palmer, M., Belvin, R., Pradhan, S., Ramshaw, L. and Xue, N. (2011). OntoNotes: A Large Training Corpus for Enhanced Processing. In Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation., (Olive, J., Christianson, C. and McCary, J., eds),. Springer.

Wiseman, S., Rush, A. M. and Shieber, S. M. (2016). Learning Global Features for Coreference Resolution. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 994–1004, Association for Computational Linguistics, San Diego, California.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., ÅĄukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K.,

Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. CoRR *abs/1609.08144*.

Xu, F., Uszkoreit, H. and Li, H. (2007). A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity. In Proceedings of ACL 2007, 45th Annual Meeting of the Association for Computational Linguistics pp. 584–591, Association for Computational Linguistics, Prague, Czech Republic.

Xu, L. and Choi, J. D. (2020). Revealing the Myth of Higher-Order Inference in Coreference Resolution. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) pp. 8527–8533, Association for Computational Linguistics, Online.

Yang, B. and Mitchell, T. M. (2016). Joint Extraction of Events and Entities within a Document Context. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 289–299, Association for Computational Linguistics, San Diego, California.

Yang, X. and Su, J. (2007). Coreference Resolution Using Semantic Relatedness Information from Automatically Discovered Patterns. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics pp. 528–535, Association for Computational Linguistics, Prague, Czech Republic.

Yang, X., Su, J., Zhou, G. and Tan, C. L. (2004). An NP-Cluster Based Approach to Coreference Resolution. In COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics pp. 226–232, Association for Computational Linguistics, Geneva, Switzerland.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. and Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Advances in Neural Information Processing Systems 32, (Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. and Garnett, R., eds), pp. 5753–5763. Curran Associates, Inc.

Yangarber, R., Grishman, R., Tapanainen, P. and Huttunen, S. (2000). Automatic Acquisition of Domain Knowledge for Information Extraction. In COLING 2000

Volume 2: The 18th International Conference on Computational Linguistics Association for Computational Linguistics.

Yao, Y., Ye, D., Li, P., Han, X., Lin, Y., Liu, Z., Liu, Z., Huang, L., Zhou, J. and Sun, M. (2019). DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics pp. 764–777, Association for Computational Linguistics, Florence, Italy.

Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics ACL '95 pp. 189–196, Association for Computational Linguistics, Stroudsburg, PA, USA.

Yu, M., Zhao, T., Dong, D., Tian, H. and Yu, D. (2013). Compound Embedding Features for Semi-supervised Learning. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 563–568, Association for Computational Linguistics, Atlanta, Georgia.

Zheng, J., Chapman, W., Crowley, R. and Savova, G. (2011). Coreference resolution: A review of general methodologies and applications in the clinical domain. Journal of biomedical informatics *44*, 1113–22.

Zhiheng, H., Marcus, T., Asli, C., Guangping, Z. and Weiqun, X. (2009). Effectively exploiting WordNet in semantic class classification for coreference resolution. In Proceedings of the Conference on Empirical Methods in Natural Language Processing Association for Computational Linguistics.

Zhong, H., Xiao, C., Tu, C., Zhang, T., Liu, Z. and Sun, M. (2020). How Does NLP Benefit Legal System: A Summary of Legal Artificial Intelligence. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics pp. 5218–5230, Association for Computational Linguistics, Online.

Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. Knowledge and Data Engineering, IEEE Transactions on *17*, 1529–1541.