Inauguraldissertation zur Erlangung der Doktorwürde der Neuphilologischen Fakultät der Ruprecht-Karls-Universität Heidelberg

Structured Representations for Coreference Resolution

vorgelegt von

Sebastian Martschat

Referent: Prof. Dr. Michael Strube
Korreferent: Prof. Dr. Simone Paolo Ponzetto
Einreichung: 5. August 2016
Disputation: 21. Februar 2017

Abstract

Coreference resolution is the task of determining which expressions in a text are used to refer to the same entity. This task is one of the most fundamental problems of natural language understanding. Inherently, coreference resolution is a structured task, as the output consists of sets of coreferring expressions. This complex structure poses several challenges since it is not clear how to account for the structure in terms of error analysis and representation.

In this thesis, we present a treatment of computational coreference resolution that accounts for the structure. Our treatment encompasses error analysis and the representation of approaches to coreference resolution. In particular, we propose two frameworks in this thesis.

The first framework deals with error analysis. We gather requirements for an appropriate error analysis method and devise a framework that considers a structured graph-based representation of the reference annotation and the system output. Error extraction is performed by constructing linguistically motivated or data-driven spanning trees for the graph-based coreference representations.

The second framework concerns the representation of approaches to coreference resolution. We show that approaches to coreference resolution can be understood as predictors of *latent structures* that are not annotated in the data. From these latent structures, the final output is derived during a post-processing step. We devise a machine learning framework for coreference resolution based on this insight. In this framework, we have a unified representation of approaches to coreference resolution. Individual approaches can be expressed as instantiations of a generic approach. We express many approaches from the literature as well as novel variants in our framework, ranging from simple pairwise classification approaches to complex entity-centric models. Using the uniform representation, we are able to analyze differences and similarities between the models transparently and in detail.

Finally, we employ the error analysis framework to perform a qualitative analysis of differences in error profiles of the models on a benchmark dataset. We trace back

differences in the error profiles to differences in the representation. Our analysis shows that a mention ranking model and a tree-based mention-entity model with left-to-right inference have the highest performance. We discuss reasons for the improved performance and analyze why more advanced approaches modeled in our framework cannot improve on these models. An implementation of the frameworks discussed in this thesis is publicly available.

Zusammenfassung

Koreferenzresolution ist eine der grundlegenden Aufgaben des automatischen Textverstehens. Die Aufgabe besteht darin zu ermitteln, welche Ausdrücke in einem Text sich auf die gleiche Entität beziehen. Koreferenzresolution ist per Definition ein strukturiertes Problem, da die Ausgabe eines Koreferenzresolutionssystems aus Mengen koreferenter Ausdrücke besteht. Aus dieser komplexen Struktur ergeben sich einige Herausforderungen, da es nicht klar ist, wie die Struktur adäquat für die Fehleranalyse und die Repräsentation von Ansätzen zur Koreferenzresolution berücksichtigt werden kann.

In dieser Doktorarbeit untersuchen wir automatische Koreferenzresolution im Hinblick darauf, wie die Struktur berücksichtigt werden kann. Hierbei widmen wir uns sowohl der Fehleranalyse, als auch der Repräsentation von Ansätzen. Insbesondere schlagen wir zwei Frameworks vor.

Das erste Framework befasst sich mit Fehleranalyse. Wir stellen zunächst Bedingungen auf, welche eine Methode zur Fehleranalyse berücksichtigen sollte. Davon ausgehend entwickeln wir ein Framework, welches auf einer strukturierten graphbasierten Repräsentation der Referenzannotation und der Ausgabe beruht. In diesem Framework werden Fehler extrahiert, indem linguistisch motivierte oder aus Daten induzierte Spannbäume der graphbasierten Repräsentationen erstellt werden.

Mit dem zweiten Framework widmen wir uns der Repräsentation von Ansätzen zur Koreferenzresolution. Wir zeigen, dass Ansätze zur Koreferenzresolution als Prädiktoren von *latenten Strukturen*, welche nicht in den Daten annotiert sind, verstanden werden können. Aus diesen latenten Strukturen wird dann in einem Nachbereitungsschritt die Ausgabe berechnet. Von dieser Erkenntnis ausgehend entwickeln wir ein Machine-Learning-Framework für Koreferenzresolution. In diesem Framework können wir verschiedene Ansätze einheitlich darstellen. Insbesondere können wir sie als Instanzen eines generischen Ansatzes auffassen. Wir stellen sowohl viele Ansätze aus der Literatur als auch neue Varianten dieser Ansätze in unserem Framework dar. Die Spannbreite der Ansatzklassen, welche wir betrachten, reicht hierbei von simplen paarweisen

Klassifikationsmethoden bis hin zu komplexen entitätsbasierten Modellen. Durch die einheitliche Repräsentation können wir Unterschiede und Gemeinsamkeiten der Ansätze transparent und detailliert analysieren.

Schließlich benutzen wir das Fehleranalyse-Framework, um einen Vergleich der Fehler verschiedener Modelle auf einem Benchmark-Korpus durchzuführen. Wir führen hierbei Unterschiede in den Fehlern auf Unterschiede in der Repräsentation zurück. Unser Vergleich zeigt, dass ein Mention-Ranking-Modell und ein Mention-Entity-Modell, welches auf Antezedentenbäumen beruht, die besten Ergebnisse liefern. Wir besprechen, wodurch diese guten Ergebnisse zustande kommen. Weiterhin analysieren wir, weshalb komplexere Ansätze die Ergebnisse nicht verbessern können. Eine Implementation der beiden Frameworks ist als Download verfügbar.

Acknowledgments

First of all, I would like to thank my supervisor, Michael Strube. His door was (almost) always open. He introduced me to computational linguistics and coreference resolution, showed me how to do interesting research, and gave me the freedom and pushed me to follow my own ideas. Thank you, Michael, for undertaking the "experiment"!

I am very grateful to Simone Paolo Ponzetto for agreeing to be the second reviewer of this thesis. I am indebted to Katja Markert for granting me the freedom to finish this thesis while already working in her group as a PostDoc.

Most work presented in this thesis was conducted while I was a PhD student at the Heidelberg Institute for Theoretical Studies (HITS). I thank the Klaus Tschira Foundation for supporting me with a PhD scholarship during most of that time. I would like to thank the people at HITS for providing a very nice and inspiring environment. In particular, I thank my colleagues from the natural language processing group: I will always think fondly of the time I could spend with you! Most of all I miss the chats with Yufang Hou and the coffee breaks with Angela Fahrni or Alex Judea. Benjamin Heinzerling, Alex Judea, Mohsen Mesgar and Nafise Moosavi deserve a special thanks for proofreading this thesis.

Lastly, I want to thank my family and friends for always being supportive. In particular, I thank Svea and Leni, who are always there for me and brighten each of my days.

Contents

1 Introduction

The aim of research in natural language understanding is to devise computational models that are able to extract meaning from a text. One of the most fundamental and popular subtasks of natural language understanding is *coreference resolution*, the task of determining which expressions in a text are used to refer to the same entity. As an example, consider the following text snippet[1].

(1) [Vicente del Bosque]$_1$ admits it will be difficult for [him]$_1$ to select [David de Gea]$_2$ in Spain's European Championship squad if [the goalkeeper]$_2$ remains on the sidelines at [Manchester United]$_3$.

[De Gea's]$_2$ long-anticipated transfer to [Real Madrid]$_4$ fell through on Monday due to miscommunication between [the Spanish club]$_4$ and [United]$_3$ and [he]$_2$ will stay at [Old Trafford]$_3$ until at least January.

We have marked all expressions that are used to refer to entities which are mentioned at least twice. Two expressions have the same index if they are used to refer to the same entity. Even in this short text, eleven such expressions appear which are used to refer to four different entities.

In order to correctly resolve the coreference relations in the text, we have to overcome two challenges. First, the task necessitates the inclusion and coordination of many different knowledge sources. For instance, the expression *he* in the last sentence is ambiguous: It could either be used to refer to VINCENTE DEL BOSQUE, or to DAVID DE GEA[2]. However, there are strong syntactic hints that DAVID DE GEA is the correct entity, for example the parallelism of *he* and *De Gea's* in the coordinated clauses. On the

[1] Taken from `http://www.theguardian.com/football/2015/sep/02/david-de-gea-manchester-united-spain-del-bosque`, accessed 11 January 2016.

[2] Throughout the thesis, we stick to the notation established in this introduction for examples: in an example, mentions are marked with square brackets with indices. When they have the same index, they are coreferent. In running text, entities are written in SMALL CAPS and expressions that are used to refer to entities are written in *italics*.

other hand, knowledge about the real world facilitates determining that *Real Madrid* and *the Spanish club* corefer.

Second, the task is a structured task, since coreference is a relation between referring expressions. Therefore, the desired output consists of sets of coreferring mentions. Hence, for an adequate modeling of the task, researchers have to account for this structural complexity.

In the research presented in this thesis we focus on the second challenge. We aim to devise a representation formalism and an analysis method for computational coreference resolution which both adequately account for the inherent structure. In the remainder of this chapter, we further motivate the research conducted in this thesis and formulate main research questions (Section 1.1), briefly summarize our contributions (Section 1.2), present the outline of the thesis (Section 1.3) and describe which parts of this thesis were published (Section 1.4).

1.1 Motivation and Research Questions

As we have described above, the objects of interest in computational coreference resolution are the sets of coreferring expressions. This suggests that models for coreference resolution should output these sets. However, from a modeling or machine learning perspective, reasoning over these sets is difficult for two reasons. First, the number of disjunctions into sets is exponential in the number of referring expressions. Hence, the search space is very large. Second, researchers have to solve the complex task of representing and scoring disjunctions of referring expressions into sets. Therefore, approaches to coreference resolution make assumptions to simplify the task. They rely on a simpler representation and then induce the disjunction into sets from this representation. In the literature we can encounter many different approaches relying on different representations. For instance, the simplest approaches cast the problem as a binary classification (coreferent or non-coreferent?) of pairs of referring expressions. More sophisticated models use structured prediction methods to determine the strongest coreference relations in a set. On the surface, these approaches tackle the task in very different manners. However, they all are models for the same task that make use of different assumptions and representations to cope with the structural complexity inherent to coreference. This observation leads to the first research question investigated in this thesis: **does there exist a *unified* representation of approaches**

to coreference resolution? If we can answer this question affirmatively, some follow-up questions and research opportunities naturally emerge. The first question is, of course, how different approaches to coreference resolution can be expressed in the representation. Based on the unified representation, we can investigate what we can deduce about modeling assumptions of the approaches and about differences and similarities between approaches. Moreover, we can study to which extent they model the structure inherent to coreference.

In order to further deepen our understanding of computational coreference resolution, we do not only need to understand the approaches on a representational level, we also need to be able to analyze and compare the output of different approaches. Both reference annotation and system output consist of disjunctions into sets of referring expressions. It is not clear from the set-based representation how to extract useful information – how can we compare these sets of sets to understand what went wrong in our prediction? This motivates our second research question: given the disjunction into sets as annotated in the data, and as predicted by the system, **what is an appropriate way to extract errors?** In order to answer this question, we first have to define an appropriate error representation. This representation should respect the structural complexity of the task and provide useful information to the developer of a coreference resolution system or a researcher working on coreference resolution.

Finally, assuming we have devised a unified representation for approaches and an appropriate method for extracting errors, we can combine these contributions to perform a large-scale qualitative analysis of approaches to coreference resolution. With this analysis, we can investigate the third main research question: **how do approaches to coreference resolution differ qualitatively and how far can these differences be attributed to the differences in the representations?** Such an analysis will enable us to assess the impact of more sophisticated representations and of modeling assumptions.

1.2 Contributions

We answer the first research question by proposing a unified representation of approaches to coreference resolution. The proposed representation is based on the insight that approaches to coreference resolution output disjunctions into sets, but their internal representation can be understood as *latent coreference structures* that encode

coreference relations. We formulate this representation as part of a machine learning framework for coreference resolution. We show how various machine learning approaches to coreference resolution can be expressed in this framework and compare the latent structures they are based on, analyzing differences, strengths, weaknesses and modeling assumptions.

We answer the second question by motivating and developing a method for error analysis of coreference resolution approaches. This method takes the structure of coreference into account by representing reference annotation and system output as graphs, and then extracts errors based on spanning trees of these graphs. We discuss various linguistically motivated and data-driven methods to compute such spanning trees.

Finally, we answer the third question by employing the error analysis method to perform an in-depth analysis of the approaches expressed in the machine learning framework on a benchmark data set. We compare models in the order of increasing expressiveness and complexity. We devise meaningful categories for the errors made by the models, and relate these errors to the structures the models operate on. We find that a mention ranking model and a tree-based mention-entity model using left-to-right inference work best. We discuss the reasons for their superior performance.

The error analysis framework, the machine learning framework and the coreference resolution models discussed in this thesis are implemented as an open source Python library[3].

1.3 Outline

The remainder of this thesis is organized into eight chapters.

In Chapter 2, we describe the task of coreference resolution in detail. We give a formal definition and discuss linguistic properties, main modeling issues and evaluation of the task.

In Chapter 3, we discuss related work in the three subfields of coreference resolution this thesis is concerned with: machine learning models, representation frameworks and error analysis.

In Chapter 4, we present our method for error analysis. We first motivate the need for error analysis and gather requirements for an error analysis method. We devise a

[3] Available for download at `http://smartschat.de/software`.

graph-based method based on spanning tree extraction following these requirements.

In Chapter 5, we review approaches to coreference resolution and observe that the approaches can be understood as *predictors of latent structures*. This observation forms the basis for a machine learning framework for coreference resolution that we present in that chapter. This framework allows for a unified representation of approaches to coreference resolution.

After having established a formal framework, we can now express approaches to coreference resolution in the framework. In Chapter 6, we demonstrate how various approaches to coreference resolution can be expressed in our framework.

Turning towards the experiments, in Chapter 7 we describe the features used by the models discussed in this thesis.

In Chapter 8, we perform a large-scale qualitative analysis of the approaches implemented in our framework. To compare the approaches we employ the analysis method presented in Chapter 4.

In Chapter 9, we summarize the answers the research in this thesis gives for the research questions formed in the motivation. We furthermore discuss avenues for future work.

1.4 Published Work

Most research presented in this thesis is an extension of published research first-authored by the author of this thesis. If not noted otherwise, the presented research is based on the contribution of the author of this thesis to the published research.

The error analysis framework presented in Chapter 4 was published in Martschat and Strube (2014) and Martschat et al. (2015b). A preliminary version of the underlying analysis method was presented in Martschat (2013). The machine learning framework presented in Chapter 5 and the discussion of structures for coreference resolution presented in Chapters 6 and 8 are based on research presented in Martschat et al. (2015a) and Martschat and Strube (2015). Our coreference resolution system, which implements the error analysis and machine learning frameworks discussed in this thesis, employs the mention extraction and mention property computation described in Martschat et al. (2012).

2 Coreference Resolution

Coreference resolution is a fundamental task in natural language processing with varying specifications over the years. In this chapter, we give a linguistic and formal definition and treatment of the task tackled in this thesis (Section 2.1). Based on the formal definition, we introduce a graph-based representation of coreference relations between mentions, which will serve as the representational foundation for the machine learning framework as well as the error analysis framework presented in this thesis (Section 2.2). Based on the formal representation and the linguistic properties, we highlight some issues that complicate modeling (Section 2.3). Lastly, we discuss the issue of coreference resolution evaluation (Section 2.4).

2.1 Problem Definition

In this thesis, we tackle the problem of noun phrase coreference resolution. The common definition of this task is to determine which noun phrases in a text are used to refer to the same entity[1]. This task has been the focus of large evaluation campaigns (MUC-6, 1995; MUC-7, 1998; NIST, 2003; Pradhan et al., 2011, 2012) and the majority of work on coreference resolution (see Section 3.1). Extensions of the task consider other constituents as potential referring expressions, such as verb phrases or clauses (Eckert and Strube, 2000; Chen et al., 2011; Kolhatkar and Hirst, 2012), or consider weaker relations than the identity of reference, which is also called *bridging* (Poesio et al., 2004; Hou et al., 2013).

2.1.1 Formal Modeling

For a formal definition of the task, we first define the meaning of *reference to the same entity*. In this regard we follow van Deemter and Kibble (2000). We first introduce terminology and notation for the linguistic expressions that are used to refer to entities.

[1]Following common terminology, we will also say that the noun phrases themselves *refer* to entities.

Definition 1. *Let d be a document and let M_d be the set of all noun phrases and pronouns in d. We call the elements of M_d* candidate referring expressions *or* mentions*.*

We order the mentions according to their position in the text. This ordering will be convenient for the presentation of the error analysis and machine learning frameworks we devise later in this thesis.

Definition 2. *We define a total ordering over the mentions in M_d as follows. $m < n$ if either m starts before n, or m and n start at the same token, but m ends before n. We write $m_1, \ldots, m_k$ for the mentions in this order.*

Mentions are used to refer to entities. We express the reference via a function $\mathrm{Referent}$ that maps mentions to the entities they refer to.

Definition 3 (Referent)**.** *For a mention $m \in M_d$, $\mathrm{Referent}(m)$ is the entity m refers to. If m does not refer to any entity, $\mathrm{Referent}(m)$ is undefined.*

The $\mathrm{Referent}$ function induces a *coreference* relation on the set of mentions.

Definition 4. *Two mentions $m, n \in M_d$ are coreferent if and only if $\mathrm{Referent}(m) = \mathrm{Referent}(n)$. We also write $\mathrm{coreferent}(m, n)$.*

This relation is reflexive, symmetric and transitive, therefore it is an *equivalence relation*. The equivalence classes contain all mentions that refer to the same entity. Given a document, our aim is to predict these equivalence classes: we do not want to determine to which entities the mentions refer to, we only are interested in whether they refer to the same entity or not.

We also need terminology to talk about two mentions referring to the same entity according to the prediction of the coreference resolution system at hand. A coreference resolution system S gets as input a set of mentions, $M_{d,S}$, and outputs assignments of mentions to entities, which we denote via a function Entity_S. Note that $M_{d,S}$ can be different from M_d. This difference can be attributed to preprocessing errors or annotation decisions. For example, in the OntoNotes annotation (Weischedel et al., 2011), non-referring noun phrases are not annotated as mentions, but most systems extract all noun phrases as mentions.

Definition 5. *For a coreference resolution system S and a mention $m \in M_{d,S}$, $\mathrm{Entity}_S(m)$ is the entity m refers to according to the system output of S. If m does not refer to any entity according to S, $\mathrm{Entity}_S(m)$ is undefined.*

Typically, since it is not part of the task, coreference resolution systems do not output an explicit representation of the entities the mentions refer to. Instead they represent the entities via arbitrary integer identifiers. However, this representation is sufficient to again induce an equivalence relation from this output.

Definition 6. *Two mentions $m, n \in M_{d,S}$ are in the same system entity according to S if* $\mathrm{Entity}_S(m) = \mathrm{Entity}_S(n)$. *We also write* $\mathrm{sameEntity}_S(m, n)$.

We can now give a formal definition of the task.

Definition 7. *The task of* coreference resolution *is as follows. Devise a system S that, given a document d, predicts the equivalence classes of the* $\mathrm{coreferent}$ *relation over M_d via the* $\mathrm{sameEntity}_S$ *relation over $M_{d,S}$.*

We also refer to the equivalence classes as *coreference chains*. Whether we refer to the classes according to the $\mathrm{coreferent}$ relation or according to the $\mathrm{sameEntity}_S$ relation will be clear from the context.

2.1.2 The Linguistics of Coreference

This formal definition is sufficient to obtain a generic representation of the objects of interest, and to develop machine learning methods. However, the definition does not relate the coreference relation to the linguistic properties of the mentions or the context in which the mentions appear. Arguably, the linguistic properties of the mentions and the context influence the processing of coreference relations between mentions. For example, when employed as referring expressions, pronouns behave differently from proper names.

Our aim is to devise adequate analysis methods and representations for approaches to coreference resolution. Therefore we need to understand what aspects in the output of an approach are useful to analyze, which features can be helpful for coreference resolution, and which factors contribute to a linguistically adequate representation of the task. To sum up, our aim presupposes a deeper understanding of the interactions between the processing of coreference relations and the linguistic context of the mentions. We now revisit the coreference relation with these considerations in mind.

2.1.2.1 Reference and Referent

In the formal definition we inferred an equivalence relation from the $\mathrm{Referent}$ assignment of mentions to entities. Hence, we established a relation between expressions

in a text solely based on a relation of these expressions to the real world. Our definition of coreference does not account for relations between the expressions that can be obtained by a linguistic analysis of the text.

In order to be able to complement our definition of coreference by a linguistic analysis, we need to introduce a model of text understanding. We adopt the popular approach based on discourse models (Webber, 1979). A text forms a *discourse*, which is a set of *utterances*. When processing a text, the hearer/reader constructs a *discourse model*, which relates linguistic expressions.

We have to distinguish between the *referent* and the *reference* of a mention (Chrystal, 2008, p. 407f.). As expressed in Definition 4, the *referent* of a mention is the entity the mention refers to. This relation is an extra-linguistic relation.

In contrast, *reference* is a phenomenon which relates mentions in the discourse model: some mentions rely on other mentions for their interpretation. More specifically, this phenomenon is called *anaphoric reference*. The mention which depends on another mention for its interpretation is called *anaphor*, the mention it refers to is called *antecedent*[2]. Anaphoric reference not only encompasses coreference. However, we restrict ourselves to anaphoric reference where the two participating mentions refer to the same entity.

2.1.2.2 Coreference versus Anaphoric Reference

From the definitions it follows that two mentions can be coreferent without being in a relationship of anaphoric reference (Fraurud, 1990, p. 406f.). Consider for example the mentions of APPLE in Example (2)[3]:

(2) Six university researchers have revealed deadly zero-day flaws in [Apple's]$_1$ iOS and OS X, claiming it is possible to crack [Apple's]$_1$ password-storing keychain, break app sandboxes, and bypass [its]$_1$ App Store security checks.

Neither of the two proper name mentions of APPLE relies on the other for its interpretation. We can observe the same phenomenon for definite common nouns, most notably in conversations, as in Example (3)[4]:

[2] A mention can refer back or forward in a discourse. If it is referring forward, it is also called an *cataphor*. We use the term *anaphor* for both cases.

[3] Taken from `http://www.theregister.co.uk/2015/06/17/apple_hosed_boffins_drop_0day_mac_ios_research_blitzkrieg/`, accessed 18 June 2015.

[4] Appears in document ch_0030, part 003 of the OntoNotes 5.0 corpus (Weischedel et al., 2013).

(3) A: Wow. With [the traffic]$_1$ and everything?
B: with yeah [the traffic]$_1$

However, the majority of nouns which are not the first in their respective coreference chain are used anaphorically. This even holds to a certain extent for proper names, such as *Mrs. Clinton* in the following example[5].

(4) [Hillary Clinton]$_1$ called for "common-sense" gun control measures and said the fatal shooting of nine people at an African-American church was not an "isolated" tragedy, but a chilling reminder of enduring racism and "bigotry" in the U.S.

In a speech Saturday at a conference of U.S. mayors, [Mrs. Clinton]$_1$, spoke extensively about the murders and praised [the victims' families]$_2$, who in court proceedings said [they]$_2$ forgave the white suspect, 21-year-old Dylann Roof.

To correctly determine the referent of *Mrs Clinton*, we need to know that it refers back to *Hillary Clinton*.

As is common in research on coreference resolution, we slightly abuse the terminology and use the terms *anaphor* and *antecedent* also for mention pairs that do not exhibit anaphoric reference:

Definition 8. *Let M_d be the set of mentions in a document d. Given two mentions $m, n \in M_d$ with $\mathrm{coreferent}(m, n)$ such that n precedes m in d, we call m the* anaphor *and n the* antecedent[6].

2.1.2.3 Reference and the Type of Referring Expression

In this thesis, we are investigating noun phrase coreference in English. When referring, writers and speakers of English can choose between three types of noun phrases: noun phrases headed by proper names, headed by common nouns and headed by pronouns. Much research in linguistics is concerned with how speakers and writers

[5] Taken from `http://blogs.wsj.com/washwire/2015/06/20/hillary-clinton-calls-for-tighter-gun-control-after-charleston-church-shooting/`, accessed 12 August 2015.

[6] In a pair (m, n), we first denote the anaphor and then the antecedent. This differs from standard terminology, where the order is vice-versa (e.g. Fernandes et al., 2014). However, this change in terminology will make the representation of the error analysis and machine learning frameworks in Chapter 4 and Chapter 5 more convenient.

choose between theses types, and how hearers and readers process them. According to Gundel et al. (1993), "It is widely recognized that the form of referring expressions [...] depends on the assumed cognitive status of the referent, i.e. on assumptions that a coöperative [sic!] speaker can reasonably make regarding the addressee's knowledge and attention state in the particular context in which the expression is used" (Gundel et al., 1993, p. 275).

Approaches differ in how they model and treat the cognitive status. For instance, Clark and Marshall (1981) claim that the choice of referring expression is governed by *mutual knowledge*. They distinguish between three types of mutual knowledge, *community membership*, *physical co-presence* and *linguistic co-presence*. For example, writers employ proper names when they can assume that the writer and the readers share the knowledge about the referent of the proper name. Linguistic co-presence on the other hand encourages the use of pronouns.

To better understand coreference, we briefly review research on how the different mention types establish reference, and how they are processed. In particular, we are interested in what the factors of choosing and comprehending the different types of noun phrases are, how they establish reference, whether they are used anaphorically or not, and how difficult it is for humans and algorithms to resolve these references.

We support our discussion by a corpus study of the coreference relation on the CoNLL-2012 training data subset (Pradhan et al., 2012) of the OntoNotes 5.0 corpus (Weischedel et al., 2013). The corpus spans several genres, from news wire to transcribed telephone conversations. We describe the corpus in more detail in Section 8.1. Table 2.1 shows an overview of the coreference properties in this corpus for proper names, common nouns and pronouns. We provide a fine-grained distinction by distinguishing between definite, indefinite, bare and other noun phrases, and by distinguishing pronouns according to their *canonical form* (for example, the canonical form of *him* is *he*). For each class of mentions, we count the total occurrences in the corpus, occurrences in coreference chains, and the proportion of mentions that are first in their chains, compared to all mentions of that type in any coreference chain.

Proper Names. *Proper names* are names of specific entities. Proper names can be used to refer to entities which writer and reader share knowledge about, like *Apple* and *Hillary Clinton* in Examples (2) and (4) respectively. However, proper names are also frequently used to introduce entities (probably) unknown to the reader, as in the

Type	Occurrences	In Chain	First in Chain	Perc. First in Chain
Proper Name	99,858	43,598	12,664	29%
Common Noun	193,930	39,189	17,041	43%
Definite	81,742	27,601	8,230	30%
Indefinite	23,560	3,062	2,634	86%
Bare	66,896	5,739	4,312	75%
Other	21,732	2,787	1,865	67%
Pronoun	78,796	66,491	2,469	4%
I	14,208	13,844	682	5%
you	13,077	9,055	351	4%
we	8,132	5,640	727	13%
he	14,625	14,468	117	1%
she	3,534	3,489	42	1%
it	12,359	8,045	203	3%
they	12,447	11,795	319	3%
this	2,780	1,334	167	13%
that	2,873	1,593	99	6%

Table 2.1: Coreference statistics for the CoNLL-2012 training portion of the OntoNotes 5.0 corpus. For each mention type total occurrences, occurrences in coreference chains and percentage of occurrences as first mention in a chain are shown.

following example[7].

(5) [Giant Group Ltd.]$_1$ said [it]$_1$ terminated negotiations for the purchase of [Aspen Airways, a Denver-based regional carrier that operates the United Express connector service under contract to UAL Corp.'s United Airlines]$_2$.

[Giant, a Beverly Hills, Calif., collection of companies that is controlled by Hollywood producer Burt Sugarman]$_1$, didn't give a reason for halting its plan to acquire [the airline]$_2$, and [Aspen]$_2$ officials couldn't be reached for comment.

The entity GIANT GROUP LTD. is introduced by the proper name *Giant Group Ltd*.

[7]Appears in document wsj_2424 from the OntoNotes 5.0 corpus.

Even if the reader is not familiar with *Giant Group Ltd.*, the reader can understand the text by inferring from the first sentence that *Giant Group Ltd.* is a company since it had plans to purchase another company. By means of an apposition, we extend our knowledge of *Giant Group Ltd* in the second sentence.

As we can see from the examples presented so far, proper names can be used to refer to known and unknown entities, and they are used anaphorically and non-anaphorically. They play an important role in introducing entities into a discourse, either by referring to known entities or by referring to unknown entities which are then described later (as in Example (5)). This assessment is also supported by the numbers in Table 2.1: from the proper names in coreference chains, about 30% are first in chain, and they constitute a large proportion of all mentions that are first in their respective chains.

Proper names are used when referring to entities that are uniquely identifiable (Gundel et al., 1993; Mulkern, 1996). However, we do not always use proper names when referring to such entities. Consider the following pair of examples, taken from Gordon et al. (1993):

(6) [Bruno]$_1$ was the bully of the neighborhood.
[Bruno]$_1$ chased Tommy all the way home from school one day.

(7) [Bruno]$_1$ was the bully of the neighborhood.
[He]$_1$ chased Tommy all the way home from school one day.

In Example (6), the entity BRUNO is referred to twice in adjacent sentences by the proper name *Bruno*. In contrast, in Example (7), the entity is introduced by the proper name *Bruno* and then referred to by the pronoun *he*. Example (7) is more readable than Example (6), due to the *repeated name penalty*: in contexts such as in the sentences displayed in the example, repeating the proper names increases reading time, which suggests that it hampers processing of the discourse (Gordon et al., 1993). Hence, the choice of whether to employ proper names or not is governed by context factors and the status of the entities in memory. In general, proper names are used to refer to entities stored in *long-term memory*, while for example pronouns are used to refer to entities stored in *short-term memory* (Ariel, 1988).

As we could already see from the examples in this section, coreference between proper names can often be resolved by simple string similarity heuristics (*Hillary Clin-*

ton and *Mrs. Clinton*; *Giant Group Ltd.* and *Giant*; ...). Many examples, however, require world knowledge, such as in the following example[8]:

(8) While the mild winters make this comfortable living for almost three million seniors, the largest elderly population in the United States, this year it is not the weather drawing [Al Gore and George W. Bush]$_1$ to [the Sunshine State]$_2$.

[Both candidates]$_1$ list 25 reasons to keep coming back, [Florida's]$_2$ 25 electoral votes.

The resolution is facilitated if we know that *the Sunshine State* is a nickname for FLORIDA.

Common Nouns. *Common nouns* are designators for classes of entities or for abstract concepts, such as *book*, *money*, *people* or *happiness*. We abuse notation and call all noun phrases common nouns which have a common noun as their head, such as *a new book* or *the happiness I pursue*.

We can distinguish these noun phrases based on their determiner. *Definite noun phrases* are noun phrases that begin with the determiners *the/this/that/these/those* or a possessive phrase, such as *the book*, *my people* or *Obama's presidency*. *Indefinite noun phrases* begin with the indefinite article *a* or *an* such as *a statue*. *Bare noun phrases* are noun phrases without any determiner, as for example *books* or *new data*. These three are the most frequent classes of noun phrases. We subsume the remaining noun phrases in an *Other* category[9].

Each of these classes serves different functions in a discourse. Definite noun phrases, which are also called *definite description*, are similar to proper names: they refer to specific entities or concepts. Most uses of the definite presuppose "the existence of the entity, set or quantity that the addressee is expected to be able to identify" (Huddleston and Pullum, 2002, p. 369). Hence, they are often used to refer to entities already introduced into the discourse. Entities can also be introduced into a discourse if the entity which is referred to is identifiable by both the writer/speaker and reader/hearer, as in the following example[10].

[8] Appears in document mnb_0010 of the OntoNotes 5.0 corpus.

[9] This a diverse category, containing nouns with determiners such as *some*, *every*, *all* and *one*; and also containing coordinated phrases. A discussion of these is out of scope of this thesis.

[10] Appears in document ectb_1070, part002 in the OntoNotes 5.0 corpus.

(9) [The devastating 21 September earthquake of 1999]$_1$ left Taiwan's landscape covered in scars, but researchers discovered that the places least damaged by [the quake]$_1$ were areas of natural forest.

Here, *the devastating 21 September earthquake of 1999* refers to a particular, identifiable earthquake, which the header/reader can identify by the detailed lexical description.

In contrast, the use of the indefinite article *a/an* does not presuppose existence or being able to identify, the addressee "is not expected to be able to identify anything" (Huddleston and Pullum, 2002, p. 371). Indefinite noun phrases play a prime role in introducing new entities into discourse, since they can express existential quantification, such as in Example (10). This property is also evident from the numbers in Table 2.1: from all uses of indefinite noun phrases in coreference chains, 86% are the first mention in the chain.

(10) I've bought a new book.

Bare plurals are often used for the same purpose as indefinite noun phrases (in English, there is no plural indefinite determiner). However, bare plurals also can often be interpreted generically, such as *Pandas* in the following example.

(11) Pandas prefer bamboo.

Here *Pandas* does not refer to a specific set of panda bears, but to *any* set of panda bears. Besides the bare plural, definite and indefinite noun phrases also permit this use for many nouns (we can also say *a/the panda prefers bamboo*). Finally, noun phrases in the *Other* category have diverse usage, which is often non-referential, such as in the following example.

(12) Every cat likes that.

Resolving coreference between common nouns is one of the most difficult subtasks of coreference resolution. To understand why, let us consider a few examples in order of increasing difficulty.

(13) [Recognition Equipment Inc.]$_1$ said [it]$_1$ settled a civil action filed against [it]$_1$ by [the federal government]$_2$ on behalf of the U.S. Postal Service.

[The government]$_2$ sued [the company]$_1$ in April, seeking $23,000 and other unspecified damages related to an alleged contract-steering scheme.

In this example[11], there is a head match between the mentions *the federal government* and *the government*. This head match provides a strong clue for coreference, which is also correct in this case. However, a head match does not always induce coreference[12]:

(14) The suit charged [the defendants]$_1$ with causing Peter E. Voss, an ex-member of the Postal Service board of governors, to accept $23,000 in bribes, kickbacks and gratuities.

[...]

The [five additional defendants]$_2$ weren't parties to the settlement .

Here, *the defendants* and *the five additional defendants* do not corefer. This can be inferred from the pre-modifier *additional*. Hence, modifier agreement is an important issue in resolving common noun coreference. Accurately modeling this agreement requires deep language understanding.

Finally, we consider mentions with no head match between the anaphor-antecedent pair[13]:

(15) In accordance with the urban plan, [new streets]$_1$ were cut this way and that through Pali's land, and high-rise buildings sprang up along the sides of [these roadways]$_1$.

To determine that *these roadways* is coreferent with *new streets*, we need to know that (at least in some contexts) *roadways* and *streets* are synonymous. Obtaining this information is difficult. For example, WordNet (Fellbaum, 1998), a standard, manually-compiled lexical database for English, does not contain the information that the nouns are in a lexical relation. In general, WordNet's coverage is too low. Distributional or embedding methods (such as Mikolov et al. (2013)), which extract co-occurrence information from large corpora, have higher coverage, but we are not aware of any successful application on top of a supervised state-of-the-art coreference resolution system.

[11] Appears in document wsj_2452 of the OntoNotes 5.0 corpus.
[12] The example appears in document wsj_2452 of the OntoNotes 5.0 corpus.
[13] The example appears in document ectb_1050 of the OntoNotes 5.0 corpus.

Pronouns. Pronouns are words that substitute for noun phrases and nouns. In this thesis, we only consider the personal pronouns *I*, *you*, *we*, *he*, *she*, *it*, *they*, and the demonstrative pronouns *this* and *that*, including all inflected forms[14].

Most pronouns are used to refer to entities already introduced into a discourse (from Table 2.1 we can see that only 4% of all pronouns are first in their respective coreference chains). They are the prime devices for anaphoric reference. For instance, returning to Example (7), it is natural to refer to BRUNO by the pronoun *he* in the second sentence, repeating the name *Bruno* makes reading more difficult.

The third-person pronouns *he*, *she*, *it* and *they* are primarily employed for anaphoric reference. The remaining personal pronouns, *I*, *you* and *we*, often are used *deictically*: the referent of the pronoun depends on contextual information, in this case the person who utters a phrase containing the pronoun. Consider the following example:

(16) A: How are [you]$_1$?

B: [I]$_1$'m fine, and [you]$_2$?

A and B engage in a conversation. The first *you* and the *I* refer to B, while the second *you* refers to A. Without the contextual information that A utters the first sentence, and that B utters the second sentence, we cannot determine the referents of the pronouns.

If we have a close look at the numbers in Table 2.1, we can see that many occurrences of *you*, *we* and *it* are not in any coreference chain. These pronouns can be divided into *generic* and *expletive* usages. Example (17) provides an instance of a generic *you*, while Example (18) provides an instance of an expletive *it*.

(17) You never know!

(18) It is raining.

In Example (17), the pronoun *you* refers to people in general (all personal pronouns permit this usage). In Example (18), *it* does not refer to a specific entity, it just fills the subject slot of the sentence and does not have any meaning. Hence, models for coreference resolution have to identify expletive use of pronouns in order to not attempt the resolution of these pronouns.

[14] We consider only these pronouns because they are the most frequent pronouns, and several other important subclasses of pronouns — such as relative pronouns — are not annotated for coreference in the OntoNotes 5.0 corpus, on which our experiments are run.

Finally, let us consider the demonstrative pronouns *this* and *that*. While their primary use is deictic, they can also be used anaphorically (Huddleston and Pullum, 2002, p. 1504f.). As we can see from Table 2.1, roughly half of all occurrences are used anaphorically and are therefore in coreference chains. In contrast to the other noun phrases and pronouns we considered so far, demonstratives frequently refer back to verb phrases, clauses or sentences (Quirk et al., 1991, p. 375). The OntoNotes annotation guidelines only permit noun phrases and verbs as mentions. Hence, if a demonstrative pronoun has a clause headed by a verb phrase as an antecedent, only the head of the verb phrase is annotated, as in the following example[15].

(19) He said [the state court]$_1$ [relied]$_2$ on the Florida Constitution to draft [its]$_1$ decision, excluding state lawmakers and [that]$_2$ violated the U.S. Constitution.

Here, *that* refers back to the fact that the state court relied on the Florida Constitution to draft its decision, and therefore *that* is annotated to be coreferent with *relied*. Since we only consider coreference resolution of noun phrases in this thesis, we do not attempt to solve such coreference relations.

In general, the resolution of the third-person gendered pronouns *he* and *she* works well with accuracy over 80% (Stoyanov et al., 2009b): there is low ambiguity, and gender identification works well for English. The other pronouns pose a greater challenge. Approaches that perform the resolution of *it* restricted to a specific dataset, such as technical manuals, obtain a good performance (Lappin and Leass, 1994), but the performance does not generalize to corpora containing diverse genres (Stoyanov et al., 2009b).

2.1.2.4 Discussion

We discussed the interactions between the processing of coreference relations and the linguistic context and properties of mentions. We mainly focused on the handling of different mention types when processing coreference relations, but also discussed issues with regard to anaphoricity and the distinction between anaphoric reference and coreference. The insights we obtained will play a fundamental role when devising adequate error representations in our error analysis framework (Chapter 4) and when devising and analyzing adequate approaches to coreference resolution and their representation (Chapters 6 and 8).

[15] Appears in document abc_0040 of the OntoNotes 5.0 corpus.

2.2 Graph-based Representations

We now build graph-based representations of coreference equivalence classes. These will be the basis for both the machine learning framework as well as the error analysis framework presented in this thesis.

The aim of coreference resolution algorithms is to predict coreference chains of the mentions in the input document d. Formally, the reference data is equivalence classes of the mentions M_d according to the $\mathrm{coreference}$ relation, while the output is equivalence classes of the system output mentions $M_{d,S}$ according to the $\mathrm{sameEntity}_S$ relation. These equivalence classes can be represented as graphs over the respective node sets. Since the relations are symmetric, they can be represented by an undirected graph, where two nodes are connected if and only if they are in the same equivalence class. We instead opt to model the relations using a *directed* graph.

Definition 9. *A directed graph is a tuple $G = (V, A)$ where V is a set and $A \subseteq V \times V$.*

Employing a directed graph has the advantage that we can conveniently represent *order*, which will make the presentation of various algorithms clearer.

Definition 10. *Let d be a document, $M_d = \{m_1, \ldots, m_n\}$. A graph $G = (M_d, A)$ represents the $\mathrm{coreference}$ relation over M_d if*

$$A = \{(m_j, m_i) \mid j > i \text{ and } \mathrm{coreferent}(m_j, m_i)\} \subseteq M_d \times M_d. \tag{2.1}$$

We write G_d for this graph. Analogously, we write G_d^S for the graph that represents the $\mathrm{sameEntity}_S$ relation over $M_{d,S}$.

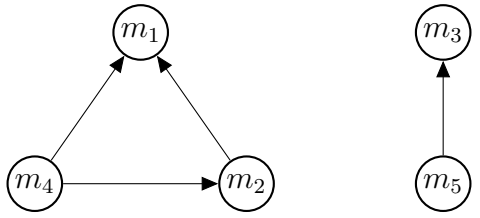

Figure 2.1: The graph G_d for a document d with mentions $M_d = \{m_1, \ldots, m_5\}$, and equivalence classes $\{m_1, m_2, m_4\}$ and $\{m_3, m_5\}$

.

Figure 2.1 shows such a graph for document d with mentions $M_d = \{m_1, \ldots, m_5\}$, where the equivalence classes are $\{m_1, m_2, m_4\}$ and $\{m_3, m_5\}$.

In these graphs, coreference chains correspond to connected components, and they model *implicitly* that a mention has no antecedent: a mention has no antecedent if it does not have any outgoing edges. However, previous work on coreference resolution found it useful to use a representation of equivalence classes which only employs one connected component and models having an antecedent *explicitly* (Chang et al., 2012; Durrett and Klein, 2013; Fernandes et al., 2014). Hence, we present an alternative representation which uses dummy mentions to make the graph weakly connected. To do so, for each node with outdegree 0, we add an edge from this node to a dummy mention m_0. Hence, in the graph that results from this procedure, the first mention of each coreference chain has an edge to the dummy mention m_0.

Definition 11. *Let d be a document, $M_d = \{m_1, \ldots, m_n\}$. We set $M_d^0 = \{m_0, m_1, \ldots, m_n\}$ with $m_0 \notin M_d$ and $m_0 < m$ for all $m \in M_d$.*

A graph $G = (M_d^0, A)$ represents the coreference *relation over M_d if*

$$A = \{(m_j, m_i) \mid j > i \text{ and } \mathrm{coreferent}(m_j, m_i)\} \cup T_0 \subseteq M_d^0 \times M_d \tag{2.2}$$

where

$$T_0 = \{(m_j, m_0) \mid \textit{ there is no } i < j \textit{ such that } \mathrm{coreferent}(m_j, m_i)\}\,. \tag{2.3}$$

We write T_d for this graph. Analogously, we write T_d^S for the corresponding graph that represents the $\mathrm{sameEntity}_S$ relation.

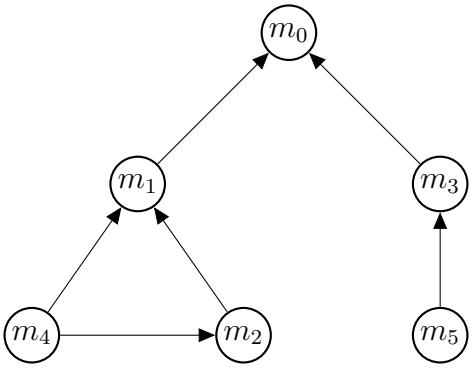

Figure 2.2: The graph T_d for a document d with mentions $M_d = \{m_1, \ldots, m_5\}$, and equivalence classes $\{m_1, m_2, m_4\}$ and $\{m_3, m_5\}$

.

Figure 2.2 shows such a graph for the example in Figure 2.1.

2.3 Modeling Issues

Machine learning approaches to coreference resolution take as input a document d. The goal is to output the equivalence classes of mentions in M_d with respect to the $\mathrm{coreference}$ relation by predicting equivalence classes over extracted mentions in $M_{d,S}$ with respect to the $\mathrm{sameEntity}_S$ relation. Typically, the set of mentions $M_{d,S}$ is obtained by a rule- or learning-based approach prior to (and independent from) the coreference resolution step. In this thesis, we assume that we have obtained $M_{d,S}$ by such a procedure. The problem is then modeled as a prediction task: given the pair $(d, M_{d,S})$, output the graph G_d^S or an equivalent representation.

As we saw in Section 2.1, predicting edges in this graph requires incorporating lexical, grammatical, discourse, context and world knowledge. Furthermore, non-trivial subtasks like anaphoricity detection or classification of expletive *it* have to be solved to correctly predict coreference chains. This multitude of tasks to be solved poses challenges for the feature designer and for the machine learning algorithm, which has to consolidate different knowledge sources and various subtasks.

Furthermore, the task is also challenging from a modeling perspective: In principal, because coreference decisions in document influence each other, we would like to train a model that predicts equivalence classes using as much information about the relations between decisions as possible. Ideally, we would like to be able to score, predict and compare different segmentations into equivalence classes directly, as this ability enables us to employ a global perspective. Hence, if we assume the graph-based representation, the machine learning approaches should learn a scoring function for graphs, and then output the highest-scoring graph that encodes the equivalence classes.

However, this approach is not feasible: the number graphs in the search space is exponential in the size of the mentions $M_{d,S}$. Therefore, machine learning approaches to coreference resolution have to resort to approximations. For instance, they obtain scores for graphs as the sum of scores for edges, and choose edges greedily. While this simplification makes the problem accessible for machine learning approaches, there is a loss of information, since they can only investigate relations for pairs of mentions.

2.4 Evaluation Metrics

For assessing the performance of models and for defining learning objectives for machine learning approaches, it is essential to have a method to evaluate the quality of a predicted set of equivalence classes. In the following, we discuss evaluation metrics for coreference resolution based on the graph-based representation described in Section 2.2.

In general, all evaluation metrics operate on the document level. For a document d, they take the equivalence classes of the $\mathrm{coreference}$ relation and the $\mathrm{sameEntity}_S$ relation as input. They then output a set of numbers evaluating the quality of the $\mathrm{sameEntity}_S$ equivalence classes compared to the $\mathrm{coreference}$ equivalence classes.

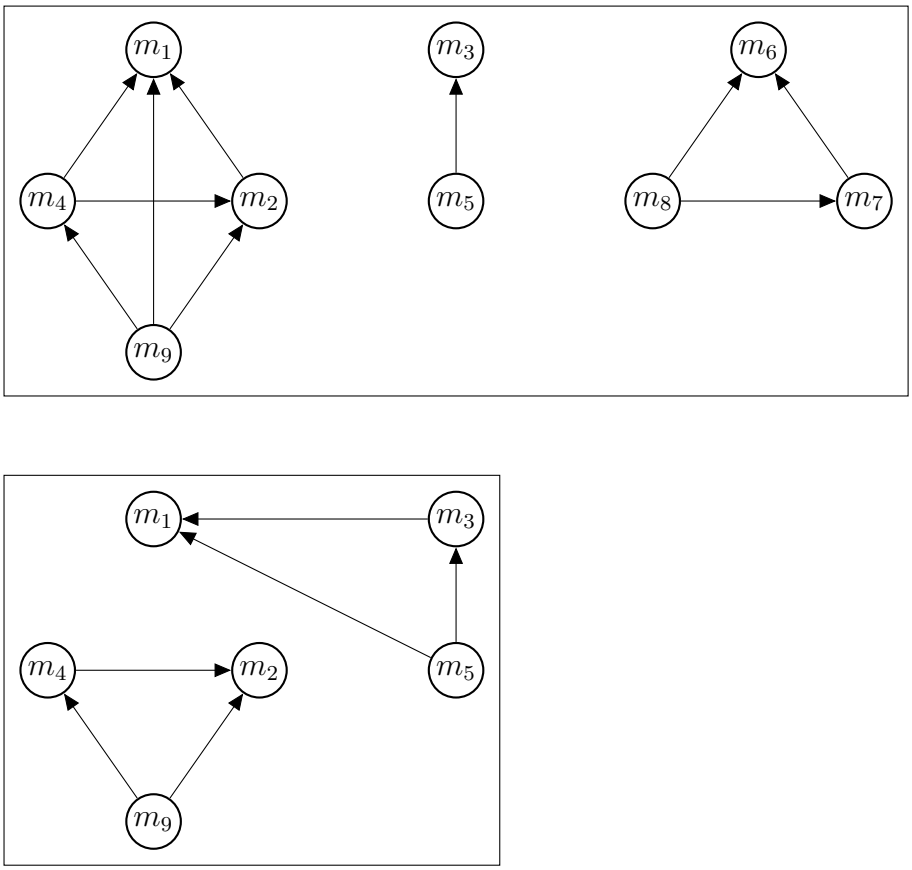

Figure 2.3: Example reference annotation (top) and system output (bottom) for coreference resolution evaluation.

2.4.1 Recall, Precision and F-Measure

All evaluation metrics for coreference resolution we consider in this thesis report performance in terms of recall, precision and F-measure. These terms originated in information retrieval, where recall measures the fraction of relevant items that were retrieved, while precision measures the fraction of retrieved items that are relevant. The F-measure combines precision and recall into a single metric by

$$F_\alpha = \frac{(\alpha^2 + 1)\, PR}{\alpha^2 P + R}, \tag{2.4}$$

where P is precision, R is recall, and $\alpha > 0$ is a parameter for controlling the recall/precision trade off. All the metrics considered in this thesis set $\alpha = 1$.

Transferring recall and precision to the evaluation of coreference resolution, recall should roughly correspond to the fraction of coreference information for reference entities that was identified correctly, while precision should roughly correspond to the fraction of coreference information that was correct in system entities.

Scoring is not a trivial task, since reference and system entities are complex objects, which can also partially match. We also want to reward when a reference entity was partially identified, or when a system entity partially corresponds to a reference entity. In the following we discuss the most popular metrics for coreference resolution, and describe how they tackle this problem. We explain the calculation of all metrics by applying the metrics to the example shown in Figure 2.3.

2.4.2 The MUC Score

The sixth Message Understanding Conference (MUC-6, 1995) introduced a coreference resolution task, which was subsequently repeated (MUC-7, 1998). Prior to these evaluation campaigns, work on reference resolution mainly considered *anaphora resolution* (the prediction of individual antecedents for mentions) and evaluated performance by counting individual links. In the coreference resolution task, where individual links are either not annotated or not of interest, we need to find a more abstract representation and evaluation metric.

Hence, Vilain et al. (1995) propose the *MUC score*, a scoring scheme that is suitable for evaluating the coreference resolution task. Their key idea is to abstract away from specific links by studying how the system output partitions G_d, and how the reference annotation partitions G_d^S. The larger the number of components in these partitions is,

the worse the output should score.

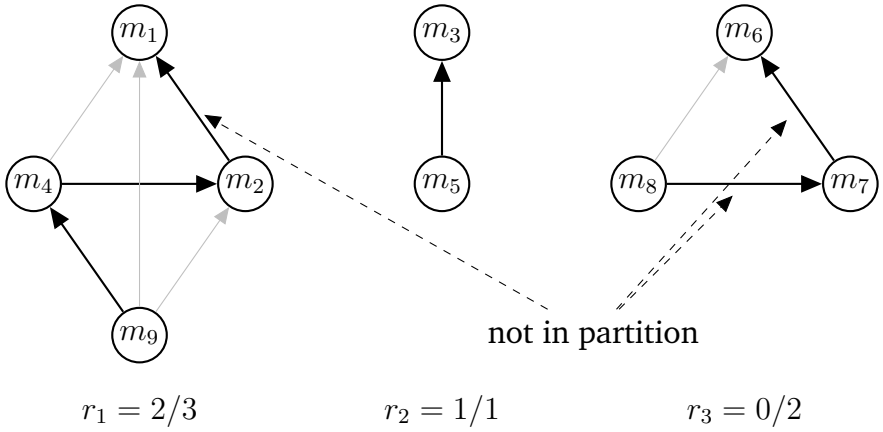

Figure 2.4: A calculation of the MUC recall score for the example in Figure 2.3. Recall is $(2 + 1 + 0)/(3 + 1 + 2) = 1/2$.

In particular, for each connected component g of G_d (corresponding to one reference entity), consider the *partition* g', which is the subgraph of g that only contains edges that can also be found in the system output G_d^S. Let t be a spanning tree of g such that t is also a spanning tree of each component in the partition. Recall g is the fraction of edges of t that are in the partition. To extend this measure to a whole document, the metric first sums over all spanning trees before computing the fraction. Figure 2.4 shows an example. For computing precision, the roles of G_d and G_d^S are switched.

2.4.3 The B^3 Algorithm

For scoring, the MUC score only considers the fraction of spanning tree edges that can not be found in the partition. In particular, the score is agnostic to the size of the individual subgraphs in the partition. Bagga and Baldwin (1998) argue that this does not adequately assess the quality of the output: conflating two large reference entities should be punished more than conflating two small reference entities.

Hence, the B^3 algorithm considers for every mention $m \in M_d$ the overlap of the reference and system entities which contain m. In terms of the graph based framework, we again start with the connected components g of G_d, and compute the partitioned graph g'. Unlike the MUC score, which is a link-based metric, the B^3 algorithm is a *mention-based* metric. Hence, for each mention m which is a node in g, it computes

the number of nodes in the connected component of g' that m belongs to. This number is then divided by the number of nodes in g.

Since each fraction for each mention is a number between 0 and 1, we extend this measure to the whole document by computing the sum over all fractions and dividing the result by the number of mentions. Figure 2.5 shows an example.

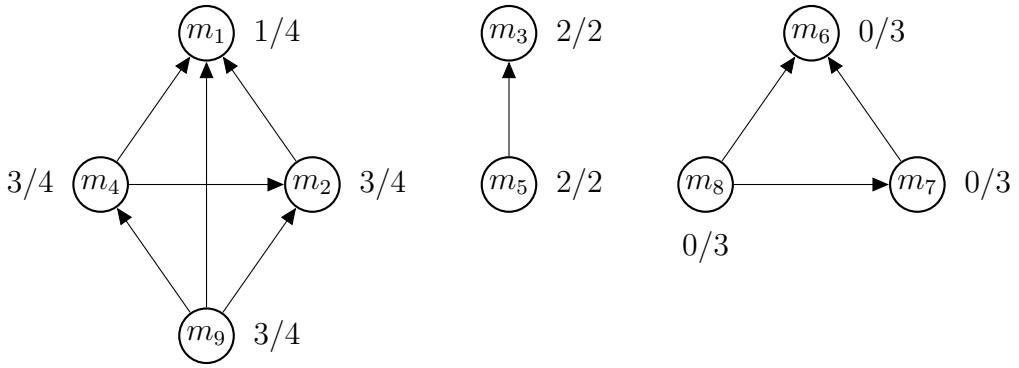

Figure 2.5: A calculation of the B^3 recall score for the example in Figure 2.3. Recall is the sum over all node scores divided by the number of mentions, which yields $(18/4)/9 = 1/2$.

For computing precision, we switch the roles of G_d and G_d^S.

2.4.4 Constrained Entity-Aligned F-Measure

Luo (2005) notes that both the MUC score and the B^3 algorithm rely on *intersections* of entities: for example, to compute recall, the metrics first compute the partition of a reference entity according to all system entities. Hence, entities can be used more than once in the computation of the metric. Luo attributes unintuitive results of the MUC score and the B^3 metric to this fact, and proposes to rely on a *bijection* between reference and system entities. In Luo's framework, we are given a similarity metric ϕ, which computes a real-valued similarity score of two entities. For a document d, we then choose an optimal bijection between the connected components in G_d and G_d^S according to ϕ. For computing recall, this number is divided by the self-similarity of G_d. For computing precision, it is divided by the self-similarity of G_d^S.

This parametrized framework gives rise to a class of metrics, which Luo calls *Constrained Entity-Aligned F-Measure* (CEAF). In his paper, Luo propose four similarity scores, where two are practical for scoring coreference resolution approaches. ϕ_3 computes the number of common mentions, where ϕ_4 computes the mention F_1-measure.

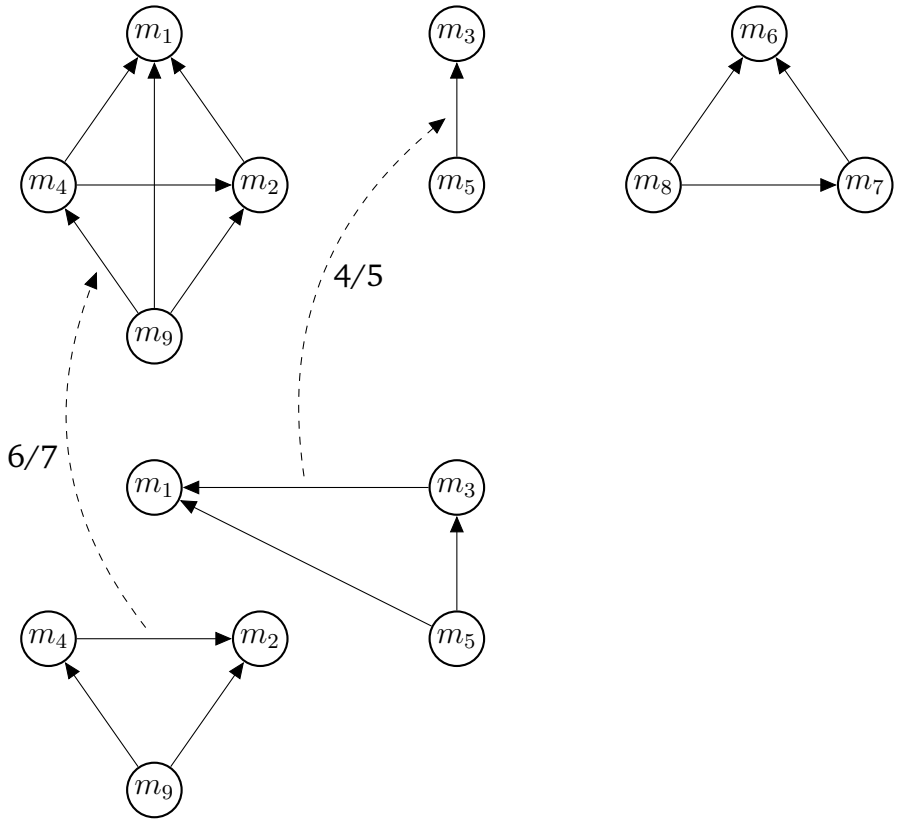

Figure 2.6: A calculation of the CEAF$_e$ score for the example in Figure 2.3. Numbers on the dashed edges are the values of the similarity function $\phi_4(K_i, R_j) = 2\,|K_i \cap R_j|\,/(|K_i| + |R_j|)$. To obtain recall, we divide $(6/7 + 4/5)$ by the self-similarity of the reference entities according to ϕ_4, which results in recall $(6/7 + 4/5)/3 = 0.55$.

Following terminology from the CoNLL shared tasks on coreference resolution (Pradhan et al., 2011, 2012), we call the former CEAF$_m$ and the latter CEAF$_e$. Figure 2.6 shows an example of the calculation of recall according to the CEAF$_e$ metric.

2.4.5 BLANC

Denis and Baldridge (2009) point out that, since the CEAF measures rely on bijections, correctly identified links can be ignored during scoring. Moreover, as Recasens and Hovy (2011) observe, on corpora where entities containing only one mention (called *singletons*) are explicitly annotated, B^3 and CEAF scores tend to be very high, which makes distinguishing between systems more difficult, since all scores are in a similar, high range.

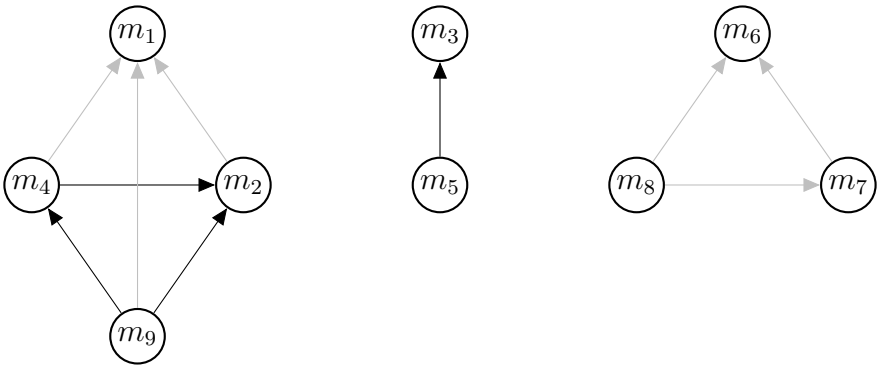

Figure 2.7: A calculation of the BLANC recall score for the example in Figure 2.3. Reference annotation and system output have four edges in common (colored black in the figure). The reference annotation contains 10 coreference edges, out of which 4 appear in the system output. Implicitly, the reference annotation contains 26 non-coreference links, out of which 24 appear in the system output (these are not displayed for readability reasons). Hence, recall is $(4/10 + 24/26)/2 = 0.66$.

Therefore, Recasens and Hovy (2011) (subsequently refined by Luo et al. (2014)) propose a new evaluation metric called BLANC (short for BiLateral Assessment of Noun-Phrase Coreference). The main idea behind BLANC is to score all *non-coreference* links in addition to all *coreference* links.

First, both are scored separately. We first discuss the case of scoring the coreference links, which correspond to edges in G_d (true links) and G_d^S (predicted links). We compute precision and recall for these links: recall is

$$R_c = \frac{\left|\text{edges}(G_d^S) \cap \text{edges}(G_d)\right|}{|\text{edges}(G_d)|}, \tag{2.5}$$

while precision is

$$P_c = \frac{\left|\text{edges}(G_d^S) \cap \text{edges}(G_d)\right|}{|\text{edges}(G_d^S)|}. \tag{2.6}$$

Additionally, we compute an F_1 score F_c for the coreference links.

Analogously, we compute precision P_n, recall R_n and F_1 score F_n for the non-coreference links, by replacing the graphs with their complement in the equations above. In order to balance the contribution of coreference links and non-coreference links, final precision, recall and F_1 score are obtained by averaging, i.e. recall is set to $(R_c + R_n)/2$, precision is set to $(P_c + P_n)/2$ and F_1 score is set to $(F_c + F_n)/2$. Figure

2.7 shows an example for computing recall according to BLANC.

2.4.6 Issues in Evaluation

The descriptions of B^3 and the CEAF metrics are underspecified with regard to situations when mentions of reference and system entities do not match (Pradhan et al., 2014). This underspecification motivated various researchers to adapt the metrics to these situations (Stoyanov et al., 2009b; Cai and Strube, 2010b; Rahman and Ng, 2011a). However, Pradhan et al. (2014) show that the original definitions of the metrics already can handle these cases, and that the proposed modifications can lead to unintuitive results.

In contrast, BLANC was devised to only handle the case where mentions in reference and system output match (Recasens and Hovy, 2011, p. 499f.). Luo et al. (2014) show how BLANC can be extended to the case where the mentions do not necessarily match.

Pradhan et al. (2014) release an open-source implementation[16] of a scoring program that implements the original definitions of the MUC score, the B^3 algorithm, and $CEAF_m$ as well as $CEAF_e$. Furthermore, it includes an implementation of the adapted version of BLANC as described in Luo et al. (2014).

[16] Available for download at `http://conll.github.io/reference-coreference-scorers/`.

3 Related Work

Since coreference resolution is a fundamental task for natural language understanding, it has received widespread attention. In this thesis, we provide frameworks for representing learning-based approaches to coreference resolution and for coreference resolution error analysis.

Accordingly, we first survey and discuss machine learning approaches to coreference resolution (Section 3.1). We then consider work on representation frameworks for coreference resolution (Section 3.2). Finally, we review work on error analysis and related fields for coreference resolution (Section 3.3).

3.1 Machine Learning Approaches to Coreference Resolution

In this section we review work on supervised machine learning approaches to coreference resolution, from simple binary classification models (Connolly et al., 1994; McCarthy and Lehnert, 1995; Aone and Bennett, 1996; Soon et al., 2001) to recently proposed structured prediction models (Chang et al., 2013; Fernandes et al., 2014; Clark and Manning, 2015).

In this thesis, we do not consider unsupervised machine learning approaches. While unsupervised coreference resolution is an active area of research (Cardie and Wagstaff, 1999; Haghighi and Klein, 2007; Ng, 2008; Poon and Domingos, 2008; Haghighi and Klein, 2010; Kobdani et al., 2011; Moosavi and Strube, 2014), supervised models constitute the state of the art and are responsible for the main innovations.

3.1.1 A Historical Overview

In order to identify trends and breakthroughs in research, we first give a brief historical overview of machine learning for coreference resolution.

Early approaches to anaphora and coreference resolution were rule-based (Hobbs, 1976; Lappin and Leass, 1994). Until the mid nineties, no sufficiently large data set annotated with coreference information was available. This absence of data made learning models from data, evaluation, and comparing approaches difficult. However, there is some work on pairwise models for coreference resolution, mainly based on decision trees (Connolly et al., 1994; McCarthy and Lehnert, 1995; Aone and Bennett, 1996). These approaches obtained competitive performance, but rule-based approaches were still prevalent.

Supervised models gained more attention after a coreference resolution task was introduced in the sixth edition of the *Message Understanding Conference* (MUC), an evaluation campaign organized by the Defense Advanced Research Projects Agency (MUC-6, 1995). The task again was evaluated in the seventh edition (MUC-7, 1998). The data sets were small (30 documents for training, and 30 for evaluation), but for the first time a standard data set and a standard evaluation metric were available.

While the entries to the evaluation campaign still were rule-based, Soon et al. (1999) and Soon et al. (2001) present a pairwise system which obtains competitive performance with a set of twelve simple features. In particular, it extracts all mention pairs from a text and classifies them using a decision tree into *coreferent* and *not coreferent*. To consolidate the decisions, the system takes the closest mention deemed as coreferent as the antecedent, which is called *closest-first clustering*. To align training instance creation with closest-first clustering, it employs heuristics to discard mention pairs during training.

The success of their simple system spurred research on machine learning for coreference resolution and established their approach as the *de facto* standard for a decade.

Therefore, much subsequent research concentrated on retaining the structure of the model and improving its components: the feature set, the learning algorithm, and the clustering algorithm for obtaining the coreference chains from the pairwise predictions.

Research was further spurred with the advent of another dataset, provided by the Automatic Content Extraction (ACE) program (NIST, 2003). Data sets for coreference resolution were released in 2002, 2003, 2004 and 2005. The ACE data sets were much larger than the MUC data sets and covered more genres. However, the annotation of coreference relations was restricted to a small set of entity types.

While the focus of research was on improving and analyzing mention pair models, few researchers worked on entity-based models, either by making use of features

that entail more than two mentions, or by considering interactions between mention pairs during learning and inference. First steps in this direction were taken by McCallum and Wellner (2003) and Luo et al. (2004). Culotta et al. (2007) achieved state-of-the-art performance, but were soon outperformed by a feature-rich mention pair model (Bengtson and Roth, 2008). Another development in coreference resolution modeling was ranking, which models the antecedent competition explicitly (Denis and Baldridge, 2008). Rahman and Ng (2011a) combine entity-based approaches and ranking by proposing a *cluster-ranking model*, which outperforms a mention pair model, a ranking model, and an entity-based model.

While large data sets were now available thanks to the ACE program, evaluating and comparing approaches was complicated: there was no reference implementation of evaluation metrics, no agreement on which data set to evaluate on, and the experimental settings (as the amount of manually annotated linguistic information used) often differed.

A third evaluation campaign, the CoNLL shared tasks on coreference resolution (Pradhan et al., 2011, 2012), finally established a standard for experimental settings, including a reference implementation of the most popular evaluation metrics (Pradhan et al., 2014).

In the CoNLL-2011 shared task, most participants employed a mention pair model in the spirit of Soon et al. (2001). However, the shared task was won by a rule-based system (Lee et al., 2011). The CoNLL-2012 shared task was won by a structured prediction approach, which modeled coreference resolution as predicting *antecedent trees*, which are trees consisting of anaphor-antecedent pairs (Fernandes et al., 2012).

The success of this model increased the interest in structured prediction models for coreference resolution, and the next years saw many ranking and antecedent tree models improving the state of the art in coreference resolution (Durrett and Klein, 2013; Chang et al., 2013; Björkelund and Kuhn, 2014; Wiseman et al., 2015).

Competitive entity-based models were devised by modeling coreference resolution explicitly as a *search problem* (Daumé III and Marcu, 2005a; Stoyanov and Eisner, 2012; Ma et al., 2014; Clark and Manning, 2015): coreference chains are built by executing a sequence of actions.

To sum up, research on coreference resolution has shifted from simple mention pair models to more complex approaches based on structured prediction. While these approaches are the most successful and widely-used approaches, entity-based models with learning and inference via search also gained a lot of attention recently.

3.1.2 Specific Models

In the overview we identified several main modeling approaches to supervised coreference resolution: mention pair models, ranking models, antecedent trees and entity-based models. We now discuss each of these categories in detail.

3.1.2.1 Mention Pair

Mention pair models have been the dominant machine learning models for coreference resolution from Soon et al. (2001) until the CoNLL shared tasks (Pradhan et al., 2011, 2012), after which structured approaches gained more attention.

Soon et al. (2001) model coreference resolution as a binary classification task for pairs of mentions. When predicting coreference chains for a document, they extract each pair of mentions (m_j, m_i) and classify it as either coreferent or non-coreferent, using a decision tree (Quinlan, 1993). To do so, they rely on twelve simple features, consisting of distance, string matching and mention type features (for example whether the anaphor is a pronoun). Their model needs a clustering step after classification, since it is there are many options to construct coreference chains from coreference decisions. Furthermore, the classification decisions can be contradicting.

Consider the following example:

(20) [Obama]$_1$ met with [Bush]$_2$. [He]$_1$ talked to [him]$_2$ for several hours.

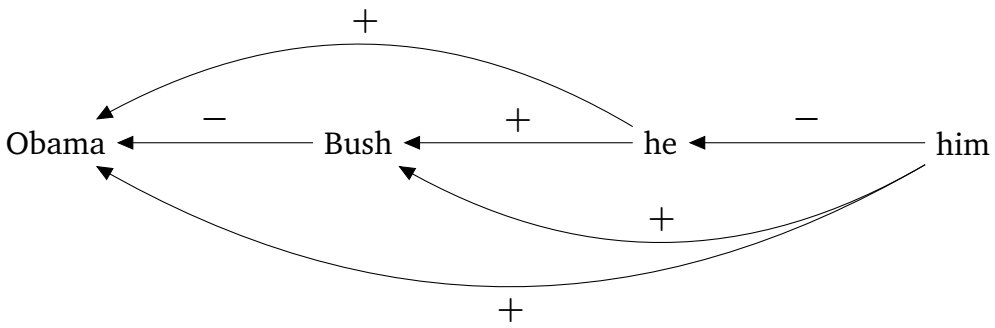

Figure 3.1: Contradicting classification decisions in the mention pair model.

Figure 3.1 shows classification decisions for the four mentions *Obama*, *Bush*, *he* and *him*. There are several contradicting decisions. For example, *he* is determined to be coreferent with both *Obama* and *Bush*, while *Obama* and *Bush* are determined to be

not coreferent. Even when ignoring the contradictions, there are several options for extracting coreference chains from the classification decisions: should we choose only one antecedent for a mention or all antecedents? If we only choose one antecedent, then which?

Soon et al. (2001) perform *closest-first* clustering to obtain the coreference chains: they choose the closest antecedent which was classified as coreferent by the model. In the example displayed in Figure 3.1, this clustering would choose *Bush* as the antecedent for *he*. Observe that the clustering does not resolve contradictions: when performing closest-first clustering, *he* and *him* are in the same coreference chain, although the model predicts that they are not coreferent.

Another issue of mention pair models is the distribution of training instances. In order to learn a mention pair model, we have to provide training data in terms of mention pairs, which are labeled as either coreferent or non-coreferent. When providing all pairs that appear in the training data, mentions may have multiple antecedents, hence the training data may not align with the clustering method that is used. Moreover, when considering all pairs, the distribution of instances is skewed, since the vast majority of pairs are not coreferent (see for example Ng and Cardie (2002)). Hence, Soon et al. (2001) employ a heuristic to change the distribution of pairs in the training data. They learn only from mentions m_j that have at least one antecedent. Let the closest such antecedent be m_i. They then consider all pairs (m_j, m_k) with $i \leq k < j$ as instances: (m_j, m_i) is a *positive* instance (the mentions are coreferent), while the remaining instances are negative.

From this discussion we can identify four parameters of the mention pair model, if we regard the underlying structure as fixed: the machine learning classifier, the feature set, the clustering algorithm, and the resampling of the training data. In the following we describe research on the mention pair model for each of these parameters.

Machine Learning Classifier. While Soon et al. (2001) used a decision tree, other research employed the perceptron (Bengtson and Roth, 2008; Stoyanov et al., 2009a), support vector machines (Rahman and Ng, 2011a), maximum entropy (Versley et al., 2008) or memory-based learning (Hoste, 2005).

Only few research systematically compares differences between these learning models. Rahman and Ng (2011a) compare maximum entropy and support vector machines, noting that differences in performance depend on the set of features used (Rahman and Ng, 2011a, p. 500).

Feature Set. The small feature set described in Soon et al. (2001) was substantially extended by Ng and Cardie (2002), who introduced 41 additional features. These additional features included more fine-grained mention type comparisons, grammatical constraints and substring match features. Subsequent research further extended this feature set. In particular, the availability of larger data sets than the MUC data used by Soon et al. (2001) and Ng and Cardie (2002) rendered the inclusion of *lexical* features feasible (Luo et al., 2004; Bengtson and Roth, 2008; Rahman and Ng, 2011a; Björkelund and Nugues, 2011).

Another import thread of research is the inclusion of *world knowledge* in coreference resolution models. Consider the following example. To correctly resolve *the US president* to *Barack Obama*, it would be helpful to provide the knowledge that Barack Obama is the President of the United States.

(21) Yesterday, Francois Hollande met with Barack Obama. The US president demanded further talks about the delicate matter.

Some of these relations can be found in WordNet (Fellbaum, 1998). Knowledge bases constructed from Wikipedia or similar resources, such as YAGO (Hoffart et al., 2011), Freebase (Bollacker et al., 2008) or WikiNet (Nastase et al., 2010) are less precise, but have higher coverage. Another option is to mine such information automatically from large unstructured datasets.

Ponzetto and Strube (2006) create a taxonomy from Wikipedia by making use of the category network (see also Strube and Ponzetto (2006)). They devise features based on semantic relatedness in this taxonomy. Rahman and Ng (2011b) devise features based on YAGO to tackle cases such as presented in Example (21), and complement these features with co-occurrence information mined from large data sets. Bansal and Klein (2012) use only unstructured text data, devising co-occurrence features from web data. All authors evaluate the usefulness of their features on the ACE data[1], and observe statistically significant improvements compared to mention pair baselines without the features.

Clustering Algorithm. The majority of work on coreference resolution has focused on improving the clustering algorithm to obtain coreference chains from the classifier's pairwise predictions. While Soon et al. (2001) always take the closest mention classified as coreferent as the antecedent, Ng and Cardie (2002) take the highest-scoring

[1]Rahman and Ng (2011b) also evaluate on OntoNotes.

mention classified as coreferent as the antecedent. This method is known as *best-first* clustering. Both of these approaches are greedy: they take a locally optimal decision and do not model any relation between decisions. Another approach that does not model relations between decisions is *aggressive-merge* clustering, which takes the transitive closure over all coreference decisions (Stoyanov et al., 2009b; Denis and Baldridge, 2009). All three of these clustering approaches have been used in various implementations, but there are few systematic comparisons. Ng and Cardie (2002) show that best-first clustering outperforms closest-first clustering on the MUC data, while Rahman and Ng (2011a) observe the reverse effect on ACE data. Denis and Baldridge (2009) compare closest-first and aggressive-merge clustering, noting that aggressive-merge improves result for the MUC metric, while closest-first gives better results for B^3 and $CEAF_m$. Stoyanov et al. (2009b) analyze the output of an resolver on MUC and ACE data, noting that for trial runs results for closest-first, best-first and aggressive-merge were comparable (Stoyanov et al., 2009b, footnote p. 658).

The clustering schemes mentioned above cannot account for dependencies between clustering decisions, such as contradictions as expressed in Figure 3.1. This spurred research on more complex clustering schemes which take such dependencies into account.

Luo et al. (2004) train a mention pair model and construct tree that represents all possible coreference chains: each node in the nth level of the tree represents a clustering of the first n mentions of the document, children of a node in the nth level are obtained by the antecedent decisions for the $(n+1)$th mention in the document. Luo et al. (2004) present an algorithm for computing optimal clusterings in the tree, which is made computationally feasible by pruning.

Nicolae and Nicolae (2006) construct a graph from the output of a pairwise classifier, where nodes are the mentions, and edges are labeled with the confidence of the classifier. They do not include pronouns in this graph. An optimal partitioning of the mentions into entities is then obtained by a variant of the MinCut algorithm (Stoer and Wagner, 1997). Sapena et al. (2013) learn a decision tree from mention pairs, and construct a graph where edge weights are derived from weights of rules expressed in the decision tree. The weight of an edge is the precision of the corresponding rule. Clusters are obtained by relaxation labeling (Hummel and Zucker, 1983).

Klenner (2007) as well as Finkel and Manning (2008) employ integer linear programming to enforce a clustering which respects the transitivity induced by the output of a pairwise classifier.

Clustering during Learning. All the approaches discussed so far only apply the clustering during inference on unseen data, not during learning. When incorporating the clustering algorithm during learning, the model does not consider each pair in isolation anymore, since decisions influence each other.

Research in this direction was pioneered by McCallum and Wellner (2003), who devised a graphical model where finding the optimal assignment of mentions to entities corresponds to graph partitioning. Chang et al. (2011) incorporate aggressive-merge clustering during learning via integer linear programming. Song et al. (2012) perform best-first clustering and a clustering which respects transitivity via Markov Logic Networks (Richardson and Domingos, 2006), an expressive formalism that combines first-order logic and Markov networks.

Training Data Resampling. Mention pair models need to cope with an unbalanced distribution of the training data labels. This is an issue since the training examples may not align with the clustering algorithm used, and the vast majority of mention pairs are not coreferent. Such skewed distributions pose problems for machine learning algorithms (He and Garcia, 2009). Hence, Soon et al. (2001) apply a heuristic to rebalance the distribution of the training data: they learn only from mentions m_j that have an antecedent. The pair of the mention m_j and its closest antecedent m_i serves as a positive instance, while all pairs (m_j, m_k) with $i < k < j$ serve as negative instances.

This heuristic was subsequently employed by the majority of work relying on the mention pair model (see, among others, Ng and Cardie (2002), Ponzetto and Strube (2006), Yang and Su (2007), Rahman and Ng (2009), and most entries to the CoNLL-2011 shared task (Pradhan et al., 2011)). Researchers often slightly modified the heuristic. Ng and Cardie (2002), for example, select the closest non-pronominal antecedent for a pronominal anaphor as the positive instance for the anaphor.

Bengtson and Roth (2008) learn their mention pair model from the closest antecedent and all preceding non-antecedents for a mention (excluding pronominal antecedents for non-pronominal mentions), while Stoyanov et al. (2009b) learn from all mention pairs.

A few studies compare different approaches for training data resampling. Hoste (2005) studies the effect of randomly downsampling negative instances on MUC data and the Dutch KNACK-2002 corpus (Hoste and De Pauw, 2006), varying the downsampling ratio. She investigates the decision tree learner Ripper (Cohen, 1995) and the memory-based learner TiMBL (Daelemans et al., 2004). According to her results,

downsampling is harmful for TiMBL but beneficial for Ripper. Recasens and Hovy (2009) and Zhekova (2011) perform similar studies for more languages, but only consider TiMBL. The results obtained by them are in line with the findings of Hoste (2005). It is worth noting that these authors only considered random downsampling and did not compare with the heuristic of Soon et al. (2001).

3.1.2.2 Ranking

The mention pair model suffers from several weaknesses. As we have identified, the output of mention pair models can contain contradicting predictions, which results in the need for additional clustering algorithms to handle these contradictions. This clustering is also not modeled during learning. Furthermore, by considering each pair individually during training, the mention pair approach is not able to model any relation or competition between different candidate antecedents for a mention.

A first step to overcoming these deficiencies was the twin-candidate model by Yang et al. (2003). In their model, for each anaphor m_j, for each each pair of candidate antecedents (m_i, m_k) they predict whether m_i or m_k is a better antecedent for m_j. The mention that wins most of these comparisons gets assigned as the antecedent for m_j. For training the model, pairs of candidate antecedents are extracted where one the mentions is coreferent with the anaphor, while the other is not coreferent with the anaphor. This approach considers all pairs of candidate antecedents during inference. However, it only considers two candidate antecedents for each individual prediction. To consider all antecedents, the approach employs a complex tournament scheme, which is also not modeled during training.

To further improve on the twin-candidate model, Denis and Baldridge (2008) propose a *ranking* approach to coreference resolution: for a given mention m_j, they consider the preceding mentions m_1 to m_{j-1} and compute scores for all pairs (m_j, m_i) with $1 \leq i \leq j-1$. The mention m_i of the highest-scoring pair (m_j, m_i) is then selecting as the antecedent of m_j. While this inference method resembles the best-first clustering used in mention pair models, the crucial difference is that a similar antecedent selection is also modeled during *training*: Denis and Baldridge (2008) learn parameter vectors such that the closest correct antecedent has a higher score compared to all other antecedents[2].

Again, we can identify parameters of the ranking approach: in contrast to the men-

[2]The set of considered antecedents is pruned according to various heuristics.

tion pair model, there exist slight structural variants for ranking models. Hence, the underlying structure is one parameter. Other parameters are the machine learning classifier to learn weights and compute scores, the feature set and the selection and handling of training data samples. Let us investigate how Denis and Baldridge (2008) and subsequent work on ranking models handled these parameters.

Structure Variants. All ranking models operate on slight variants of the same underlying structure: a mention m_j and the list of its candidate antecedents m_1 to m_{j-1}. Denis and Baldridge (2008) work on this structure, and enforce that every mention considered by their ranking model must be resolved to some antecedent. However, most mentions are not anaphoric. To handle these mentions, Denis and Baldridge (2008) first apply an anaphoricity classifier to determine whether the mention under consideration is anaphoric or not. If the mention is deemed to be non-anaphoric, it is not considered by the ranking model and left unresolved. Rahman and Ng (2011a) employ the same strategy.

Chang et al. (2012) introduce a *dummy mention* m_0, which they add to the list of candidate antecedents for each mention. If m_0 is chosen as the antecedent, the mention in focus is considered as non-anaphoric. Therefore, Chang et al. (2012) do not need an additional anaphoricity classifier.

Machine Learning Classifier. Similar to the mention pair model, ranking models were devised using many different machine learning classifiers. Denis and Baldridge (2008) employ a maximum entropy classifier, while Rahman and Ng (2011a) use support vector machines. Chang et al. (2012) train their ranker via an averaged perceptron, while Durrett and Klein (2013) employ AdaGrad (Duchi et al., 2011), a variant of stochastic gradient descent.

Feature Set. Many mention ranking models use features devised for mention pair models such as Ng and Cardie (2002) and Bengtson and Roth (2008). Durrett and Klein (2013) show that a mention ranking model can perform very well when mainly relying on lexical features and a linguistically motivated heuristic for devising feature conjunctions. Wiseman et al. (2015) automatically learn feature combinations via neural networks, which yields improvements over the heuristic combination scheme from Durrett and Klein (2013).

Chang et al. (2012) do not compute any features when considering a pair which includes the dummy mention. Therefore every pair that contains the dummy mention receives the score 0. In contrast, Durrett and Klein (2013) introduce a feature *antecedent=NEW*, which is triggered when the antecedent is the dummy mention, and conjoin this feature with features of the anaphor.

Selection and Handling of Training Data Samples. Similar to the mention pair model, some mention ranking approaches filter the training data. Denis and Baldridge (2008) learn only from anaphoric mentions, and choose the closest correct antecedent as the reference antecedent for pronouns, while they choose the closest non-pronominal antecedent as the reference antecedent for non-pronominal mentions. Work that uses dummy mentions, such as Chang et al. (2012) and Durrett and Klein (2013), considers all mentions to resolve, but usually makes use of *cost functions* to guide learning. These cost functions allow to inject knowledge of the severity of different errors. Chang et al. (2012) use a simple 0-1 loss. Durrett and Klein (2013) devise a cost function that distinguishes between three types of errors, which are weighted differently:

- a *wrong link* error happens when two non-coreferent non-dummy mentions are assigned to the same entity;
- a *false new* error happens when a mention gets the dummy mention as antecedent, but is anaphoric;
- a *false anaphoric* error happens when a mention gets a non-dummy mention as antecedent, but is not anaphoric.

Another difference between various instances of the ranking approach is the choice of reference antecedents during training. Denis and Baldridge (2008) choose the antecedents according to a heuristic: for each mention, the closest correct antecedent of a specific mention type is regarded as the reference antecedent. Hence, models following Denis and Baldridge (2008) (such as Rahman and Ng (2011a)) employ *fixed antecedents*.

These fixed antecedents do not necessarily provide the best instances for training. For instance, a common noun antecedent for a proper name anaphor may be hard to resolve. While we could devise further rules to choose other antecedents for these cases, this approach would necessitate a large amount of engineering, and it is furthermore language-, corpus- and genre-specific. An alternative is to *learn* what the

best correct antecedent for a mention is. This is the approach taken by Chang et al. (2012) and Durrett and Klein (2013): learning works in an online fashion. At each step, when considering a mention m_j, the approach chooses the highest-scoring correct antecedent under the current model. This antecedent then serves as the reference antecedent.

3.1.2.3 Antecedent Trees

Antecedent trees gained a lot of attention after Fernandes et al. (2012), who employed an model based on this structure, won the CoNLL-2012 shared task on multilingual coreference resolution (Pradhan et al., 2012). The structure was initially proposed by Yu and Joachims (2009). Antecedent trees extend the ranking approach by providing a document-level perspective: while the ranking approach considers only one anaphor-antecedent pair at each inference and training step, antecedent trees encode all anaphor-antecedent decisions for the whole document. The goal is to predict a tree with the dummy mention as root, and where each edge is an anaphor-antecedent pair (if no dummy mention is used, a spanning forest is predicted instead). The structure is a tree because each mention is restricted to have only one antecedent (which can be the dummy mention).

It is worth noting that some approaches devised as ranking perform learning via a variant of stochastic gradient descent, and perform parameter updates after considering document-size mini-batches. Since these mini-batches contain antecedent decisions for the whole document, the approaches predict antecedent trees during learning (Durrett and Klein, 2013; Wiseman et al., 2015).

Analogously to the ranking models, we can distinguish between the following parameters for antecedent trees: the exact underlying structure, the machine learning classifier to learn weights and compute scores, the feature set and the selection and handling of training data samples.

Structure Variants. Yu and Joachims (2009), who originally proposed the approach, do not employ dummy mentions. Therefore, their output does not consist of a tree, but of a set of trees, where each tree corresponds to one entity. Moreover, their graph is undirected. In practice, however, this makes no difference, since edges always represent anaphor-antecedent decisions.

Fernandes et al. (2012, 2014) employ dummy mentions. The dummy mention is the

root of the tree, edges are directed and are from antecedent to anaphor. Each subtree of the root node corresponds to an entity. All remaining antecedent tree approaches we are aware of build upon the representation of Fernandes et al. (2012, 2014) (Chang et al. (2013); Björkelund and Kuhn (2014); Lassalle and Denis (2015)).

Machine Learning Classifier. The most popular machine learning approach is the latent structured perceptron (Collins, 2002; Sun et al., 2009) or a variant thereof. It is employed by Fernandes et al. (2014), Björkelund and Kuhn (2014) and Lassalle and Denis (2015). Yu and Joachims (2009) use latent structural support vector machines; Chang et al. (2013) employ a variant of stochastic gradient descent.

Feature Set. Antecedent tree approaches use a rich feature set (including lexical features), heavily relying on feature combinations. Most approaches do not compute any features when the antecedent is the dummy mention. The exception is Lassalle and Denis (2015), who devise features for anaphoricity detecion which they apply when the antecedent is the dummy mention. They report improvements when evaluating only on mentions that can be found in the reference annotation.

Chang et al. (2013) and Lassalle and Denis (2015) apply must-link and cannot-link constraints, such as enforcing the model to build an edge between two identical proper nouns. Lassalle and Denis (2015) apply them also during learning, Chang et al. (2013) only during inference. Chang et al. (2013) observe improvements, Lassalle and Denis (2015), however, observe a decrease in performance.

In most models, the feature function is required to factor with respect to the edges in the graph, which restricts the models to features over pairs of mentions. Björkelund and Kuhn (2014) also allow *non-local* features, that apply to more than one mention, such as the sequence of mention types in an entity. They perform learning and inference using left-to-right decoding (in document order) via beam search.

Selection and Handling of Training Data. Approaches based on antecedent trees do not employ any resampling, but use cost functions to guide learning.

To compute the loss for a document, Yu and Joachims (2009) reward edges linking coreferent mentions by 1, and penalize linking two non-coreferent edges by -1[3]. The

[3]These values, as well as the values for the remaining cost functions discussed, are later scaled by a parameter which was optimized on development data or by cross-validation.

resulting value is subtracted from the number of reference mentions and entities in the desired output.

Fernandes et al. (2014) use a simpler cost function: edges between two non-coreferent mentions have cost 1, erroneous links to the dummy mention have cost 1.5, Björkelund and Kuhn (2014) use the same cost function. Chang et al. (2013) and Lassalle and Denis (2015) penalize all wrong links with cost 1.

3.1.2.4 Entity-based Approaches

All the models discussed so far (with the exception of Björkelund and Kuhn (2014)) make all decisions based on scores between pairs of mentions. Hence, they are unable to exploit features that apply to more than two mentions.

However, such features may be helpful for coreference resolution. For example (inspired by an example discussed by McCallum and Wellner (2003)), consider a document that includes the mentions *Barack Obama*, *Obama* and *she*, where only the first two are coreferent. If *she* is very close to *Obama*, likely *she* will erroneously be resolved to *Obama*. If we add a cluster-level gender agreement feature, this error could have been prevented.

While sophisticated clustering algorithms for the mention pair model can handle such cases, incorporating cluster-level features gives more expressiveness. The first such entity-based model was introduced by Luo et al. (2004) who devise an *mention-entity model*, which decides to which partially constructed entity an anaphor should be attached. Culotta et al. (2007) develop a more expressive model which can incorporate features between pairs of entities. Following the literature (Lee et al., 2013; Clark and Manning, 2015), we call such models *entity-centric*.

In the following, we discuss entity-based approaches, again according to four dimensions: the structure, the machine learning classifier, the feature set and the selection and handling of training data.

Structure Variants. Mention-entity models (Luo et al., 2004; Daumé III and Marcu, 2005a; Daumé III, 2006; Yang et al., 2008; Rahman and Ng, 2011a; Webster and Curran, 2014; Ma et al., 2014) go through the document in a left-to-right fashion, and decide for each mention to which partially constructed entity it should be attached. Hence, they can harness information about the mention in focus and any partially constructed entity so far. Most mention-entity models consider each pair of mention

and partial entity in isolation during training: they either constitute a positive instance or not. Rahman and Ng (2011a) propose a *cluster-ranking* model, which also models the search for the best partially constructed entity during search, by performing a ranking over these entities. Similarly, inspired by shift-reduce parsing (Aho and Johnson, 1974), Webster and Curran (2014) search for the best partially constructed entity on a stack.

Entity-centric models (Culotta et al., 2007; Stoyanov and Eisner, 2012; Clark and Manning, 2015; Wiseman et al., 2016; Clark and Manning, 2016) instead construct the entities via *merge* operations for pairs of partially constructed entities. This enables the approaches to take into account information about both entities, which makes these approaches more expressive than the mention-entity models.

Machine Learning Classifier. Luo et al. (2004), Yang et al. (2008) and Rahman and Ng (2011a) regard the problem as a classification task (or ranking in the case of cluster-ranking). Hence, predictions are performed by a standard classification model, such as maximum entropy or support vector machines. Daumé III and Marcu (2005a), Daumé III (2006), Webster and Curran (2014) and Ma et al. (2014) regard constructing entities in the mention-entity model as a search problem. Daumé III and Marcu (2005a), Daumé III (2006) and Webster and Curran (2014) learn parameters using a variant of the perceptron, Ma et al. (2014) split the problem into learning how to prune and score actions during search, both pruning and scoring are considered as rank learning.

Similar to Daumé III and Marcu (2005a), Daumé III (2006) and Webster and Curran (2014), all entity-centric approaches model the task as a search over merge operations. Learning therefore consists in finding parameters for determining good merge operations. To learn these parameters, the approaches mimic test-time prediction during learning. To update parameters, they employ *imitation learning*: updates are based on comparison to actions which are examples of good merge operations.

Feature Set. Most mention-entity models extend features from the mention pair model to the entity-level via logical predicates. For example, let us consider the pairwise head match feature. Luo et al. (2004) trigger the entity-based version of this feature if there is an head match between the mention in focus and any mention in the partially constructed entity under consideration. Distance features are extended by taking the minimum distance between the mention in focus and any mention in the

entity.

In their entity-centric approach, Culotta et al. (2007) use an additional set of logical predicates, by triggering features when the pairwise feature holds for none, all, or the majority of pairs obtained by pairing the mention in focus and the mentions in the entity. They also include cluster size and the output scores of a pairwise model as features. Clark and Manning (2015) devise more features based on the output of pairwise models, such as the average probability of coreference between mentions in two clusters. Wiseman et al. (2016) and Clark and Manning (2016) learn feature representations of entities automatically via neural networks.

Selection and Handling of Training Data. The entity-centric approaches and the search-based mention-entity approaches learn from the training data by constructing clusters from scratch. In each step, the predicted merging decision is compared with a good merging decision. For example, Stoyanov and Eisner (2012) consider as good merging decisions all decisions that lead to an increase of the evaluation metric they optimize for.

In contrast, Luo et al. (2004) in their mention-entity model represent coreference decisions using a *Bell* tree: the ith level of the tree contains as nodes all possible partitions of the first i mentions in the document into clusters. There is an edge between a node on the ith level and the $(i+1)$th level if the clustering on the $(i+1)$th level extends the clustering from the ith level. Hence, a node-edge pair represents an assignment of a mention to a partially constructed entity. Luo et al. (2004) take each such assignment on the path through the tree which only includes clusters consistent with the reference annotation as positive examples, and all assignments emitting from nodes on this path as negative examples. Yang et al. (2008) adopt the standard training strategy for the mention pair model to their mention-entity model: the positive examples are the same as is Luo et al. (2004). Negative examples are constructed by pairing the mention m_j in focus with the entity of each of m_j's preceding mentions until the first correct antecedent of m_j is encountered. For their cluster-ranking approach, Rahman and Ng (2011a) pair a mention with each of the preceding partial entities.

3.1.3 Discussion

Coreference resolution has been tackled by a wealth of machine learning approaches, from binary classification to incremental structured prediction. In this thesis, we extend previous work from two perspectives: first, we provide a machine learning framework that allows for a *unified representation* of approaches to coreference resolution. We show how the approaches discussion in this chapter fit into the framework and perform a large-scale evaluation and analysis. Second, we devise new variants of the approaches in the framework.

When devising the framework, we rely on the observations by Chang et al. (2012) and Fernandes et al. (2012, 2014) that coreference resolution can be modeled as predicting latent antecedents or latent antecedent trees. We note that also other approaches can be regarded as predicting such latent structures, which therefore leads to a general framework for coreference resolution. We also draw on the repeated successful use in coreference resolution of learning to search (Daumé III, 2006; Clark and Manning, 2015, inter alia) and of variants of the perceptron algorithm for learning parameters (Bengtson and Roth, 2008; Stoyanov and Eisner, 2012; Fernandes et al., 2014; Björkelund and Kuhn, 2014, inter alia).

3.2 Representation Frameworks for Coreference Resolution

Only few research on coreference resolution explicitly devises frameworks to compare individual approaches. Most work compares the proposed approach to a simple mention pair baseline modeled after Soon et al. (2001) or Ng and Cardie (2002) and/or compares to state-of-the-art systems from the recent literature, but without any deep analysis (Cai and Strube, 2010a; Durrett and Klein, 2013, inter alia).

Rahman and Ng (2011a) propose an *entity ranking* model (called *cluster ranking* by the authors), which, for each anaphor, ranks the preceding partial entities. To put their approach into context, they also implement a mention pair model, a mention ranking model and an entity mention model. They extensively discuss the differences in the learning objectives for pair classification and ranking models: the former output a class value for each pair, while the latter output a *rank* for each pair. They then provide an evaluation and comparison of the different approaches on the ACE 2005

data (Walker et al., 2006) by varying

- the feature set (non-lexical, lexical and combined),
- the machine learning classifier (support vector machines and maximum entropy), and
- the incorporation of anaphoricity determination (joint vs. pipelined).

They find that their entity ranking model with the combined feature set, based on support vector machines and joint anaphoricity determination performs best. However, the mention ranking model also performs competitively.

The work presented in this thesis differs from Rahman and Ng (2011a) in several fundamental points. First, while Rahman and Ng (2011a) do not develop a common representation for different approaches, we propose a graph-based framework for uniformly representing approaches to coreference resolution. Second, Rahman and Ng (2011a) perform a quantitative evaluation and comparison of the approaches in terms of B^3 and CEAF scores. We complement such a quantitative approach with a qualitative analysis based on the error analysis method described in Chapter 4.

3.3 Error Analysis and Related Topics

Methods for error analysis are useful to identify weaknesses of an approach, as well as to compare the output of different approaches qualitatively. Therefore, a lot of work on coreference resolution complements the reporting of the metric scores by an analysis of the errors made.

This analysis can be performed across various dimensions. Some researchers also evaluate only on subsets of the corpus (Ng and Cardie, 2002; Stoyanov et al., 2009b; Chang et al., 2013), for example on all pairs of proper nouns. This gives an estimate of the performance when dealing only with specific phenomena. In other work, the authors choose a random set of errors which they then analyze in detail (Soon et al., 2001; Lee et al., 2013). However, only few work explicitly deals with error analysis for coreference resolution. In this section, we discuss these proposals.

3.3.1 Error Analysis

While the analysis methods mentioned above can provide valuable insights, they have several deficiencies: even a fine-grained computation of metrics for different error classes can only provide a limited view on the individual errors. Closely analyzing a random subset of errors can provide information about individual errors, but the representativeness is unclear. Besides our own proposal, which we describe in detail in Chapter 4, we are aware of two other proposed frameworks for error analysis in coreference resolution.

Uryupina (Uryupina, 2007, 2008) devises a state-of-the-art mention pair model with closest-first clustering and performs an in-depth error analysis of this model. To do so, she extracts all recall and precision errors made by the system on the MUC-7 test data, which consists of 20 documents. For recall errors, she takes the "intuitively easiest" link missed by the system to analyze as a recall error (Uryupina, 2007, p. 196). For precision errors, she considers all erroneous pairwise links returned by the system.

Her in-depth analysis of these errors shows that few require deep knowledge which can not be provided by a corpus-based algorithms. The majority, however, could be resolved by such an algorithm. She sketches directions for further work, such as investigating more global resolution strategies than the clustering of pairs employed by her model.

Similarly to Uryupina (2007, 2008) our error analysis framework is based on *links* between mentions. In Uryupina's framework, she does not give a formal criterion for extracting missing links for recall errors, which makes a large-scale analysis difficult. In contrast, we give a formal framework where different notions of errors can be modeled via spanning tree algorithms. We will present spanning tree algorithms that aim to model the "easiness" of links. Our framework also can extract links for non-pairwise systems, a case which Uryupina (2007, 2008) does not consider. Furthermore, while Uryupina (2007, 2008) only analyzes one system, we use our error analysis method to compare multiple systems in detail.

Kummerfeld and Klein (2013) devise a formal framework for error analysis and apply the framework to analyze and compare a large number of different approaches. Their method is based on extracting errors from transformations of system entities to reference entities. They define a set of valid transformation operations, for example altering the span of a mention, introducing a mention, or merging two entities. A transformation or a sequence of transformations then corresponds to errors. For ex-

ample, an altering of a span corresponds to a *span error*, while introducing a mention followed by a merge operation involving this mention corresponds to a *missing mention* error. They apply their method to analyze the errors of several publicly available systems and of the task participants on the CoNLL-2011 test data.

In their analysis, they first compare the errors of all the systems, noting that "there is considerable variability in the distribution of errors, and the best systems are not best across all error types" (Kummerfeld and Klein, 2013, p. 274). Moreover, no system was able to address recall-related errors effectively. They then average the errors over the ten best-performing systems and consider the most frequent errors of each error type. Here, no error type is particularly outstanding.

In contrast to Kummerfeld and Klein (2013), who use a *transformation-based* approach, the error analysis framework we propose is *link-based*: each error made by the system is represented as a spurious or missing link. This mirrors the paradigm behind approaches to coreference resolution, even entity-centric approaches mainly rely on links between pairs of mentions (Stoyanov and Eisner, 2012; Clark and Manning, 2015). Moreover, the way we apply our analysis framework in this thesis is different from the the way Kummerfeld and Klein (2013) apply their analysis framework: in Chapter 6, we will employ our analysis framework to analyze differences and common errors between individual approaches. Kummerfeld and Klein (2013), however, apply their method to aggregate errors over top performing system. Hence, they do not analyze differences between systems in detail. The analyses provided in this work and by Kummerfeld and Klein (2013) complement each other.

3.3.2 Evaluation Metrics

For coreference resolution error analysis, we are interested in *extracting* recall and precision errors, which are then subject to further analysis. In contrast, coreference resolution evaluation metrics *quantify* recall and precision errors. This is a fundamental difference, since evaluation metrics do not provide any means to extract errors. Hence, error analysis methods complement insights that can be obtained via evaluation metrics. However, since both error analysis and evaluation metrics deal with errors, they are closely related. We discuss differences and similarities of our error analysis framework compared to evaluation metrics after giving the details of our analysis framework in Section 4.5.

4 A Method for Link-based Error Analysis

In this chapter we motivate and devise a method for extracting errors of a coreference resolution system compared to a reference corpus. We first motivate the need for error analysis methods (Section 4.1) and formulate desiderata for such methods (Section 4.2). We then present a generic algorithm for coreference resolution error extraction based on spanning trees (Section 4.3) and discuss several spanning tree variants for extracting recall and precision errors (Section 4.4). We conclude the chapter by analyzing the relation of the proposed method to coreference resolution evaluation metrics (Section 4.5).

4.1 Why Error Analysis?

Coreference resolution is a fundamental task in natural language processing, with applications in fields such as summarization (Steinberger et al., 2007), machine translation (Hardmeier et al., 2013) and question answering (Morton, 2000). Applying coreference resolution to such real-world tasks requires *end-to-end coreference resolution*, which does not assume any manually annotated linguistic information (as for example named entity classes or mention spans) as input. State-of-the-art approaches to end-to-end coreference resolution achieve a performance in the mid fifties to low seventies, depending on the evaluation metric used (Clark and Manning, 2015; Wiseman et al., 2015). Hence, to facilitate system development, and to better understand challenges in coreference resolution, we need methods for error analysis.

For some tasks designing an error analysis method is comparatively easy. For example, in a sequence labeling task we can take all the elements of the sequence which got assigned an incorrect label. However, coreference resolution is a *set-based* problem: the expected output as well as the reference annotation consists of a mapping of

mentions to entity identifiers, which can equivalently be regarded as an assignment of mentions to sets. This makes even the question of what an *error* is difficult to answer. Given two mappings e_{system} and $e_{\text{reference}}$ of mentions to entity identifiers, where e_{system} is the system output and $e_{\text{reference}}$ is the reference annotation, what should be the errors if the mappings do not match?

To investigate this further, let us consider as a simple example a document with mentions m_1, m_2, m_3, m_4 and m_5. Assume that the reference annotation is $\{m_1, m_2, m_5\}$ and $\{m_3, m_4\}$, while the system output consists of the sets $\{m_1, m_2, m_3\}$ and $\{m_4, m_5\}$. It is not clear how to measure the difference between the two set assignments: on the one hand, one could argue that m_3 and m_5 belong to the "wrong" sets in the system output, on the other hand one could take the position that instead the assignments of m_3 and m_4 to the sets are erroneous. Moreover, even if we would agree on how to measure the difference, it is still unclear how we can make this information useful for developing coreference resolution systems. Just stating that the mismatch is caused by m_3 and m_5 belonging to the wrong sets will not help a developer of a coreference resolution system, since better understanding the cause of the error will require too much manual intervention, rendering a large-scale analysis impossible. We therefore need specialized methods for error analysis that address these problems.

4.2 Desiderata

The previous section described why we need error analysis methods for coreference resolution. Furthermore, by an analysis of the problem we arrived at a few desiderata a method for coreference resolution error analysis should have. It should be able to

- cope with the set-based nature of the task by only requiring sets describing the system output and the reference annotation as inputs;
- represent errors such that a system developer or researcher can directly work with these errors to improve the system or obtain insights;
- allow for different notions of errors, depending on the underlying task (for example research on challenges in coreference resolution, or improving a system for a particular task).

In the following, we describe each of these desiderata in more detail.

Coping with the Set-based Nature. Coreference resolution systems output sets of mentions, and the reference annotation also consists of sets of mention. While most systems output anaphor-antecedent pairs and some corpora have explicit links annotated (for example the MUC corpora (Chinchor and Sundheim, 2003; Chinchor, 2001)), an error analysis method should work for the general case where just the sets are provided. However, it should be able to account for special cases where link-based information is available.

When dealing with the general set-based case, the method should be based on a formally well-founded algorithm for extracting errors from the differences between sets in the system output and in the reference annotation.

Useful Error Representation. Given system output and reference annotations for a corpus, the error analysis method outputs a set of errors. Many different error representations are conceivable. For example, building on the set-based nature of coreference resolution, we could represent entities in the reference annotation and the system output as *sets*, and represent errors as *set operations*. This is the approach taken by Kummerfeld and Klein (2013). One example of an error in their framework is *divided entity*. Such an error happens if a set of coreferent mentions in the reference annotation can be obtained by merging two sets from the system output.

When a user wants to improve a system with respect to the errors made by the system, the user still needs to analyze the set-based errors further in order to know where to improve the system. The difficulty of analyzing increases the larger the sets under consideration are. Furthermore, virtually all systems do not operate in a set-based environment. This is obvious for mention pair, ranking and antecedent tree models, but also holds for many entity-centric approaches: even these rely mainly on pairwise links between mentions (Stoyanov and Eisner, 2012; Clark and Manning, 2015).

This paradigm suggests a *link-based* representation: each error, either recall or precision, is represented as a link between two mentions. Such a representation enables the developer or researcher to directly know where to improve the system, and also aligns with the pairwise link paradigm behind approaches to coreference resolution.

Flexibility in Notion of Error. Different tasks necessitate different notions of errors. In general, we may consider any antecedent of a pronoun as equally important. Tuggener (2014) argues that for machine translation, nominal antecedents of pro-

nouns are more important than pronoun antecedents when nouns have grammatical gender, since this information helps translation.

Hence, the error analysis method should be able to accommodate for different notions of errors, depending on the task the user or researcher is interested in.

4.3 A Spanning Tree Algorithm for Error Extraction

We now develop a framework for error analysis that accounts for the desiderata presented in the previous section. We will motivate and explain the framework with help of the following running example:

(22) After the discussion, [Obama]$_1$ confirmed [he]$_1$ will return. Then [the president]$_1$ and [his]$_1$ bodyguards left.

The only non-singleton entity in this example is the BARACK OBAMA entity, and it is referred to by the mentions {*Obama*, *he*, *the president*, *his*}. Now suppose a coreference resolution system outputs the entities {*Obama*, *he*} and {*the president*, *his*} (if not equipped with world knowledge, this is the output to be expected from most systems). Intuitively, the system made a recall error: not all mentions which are coreferent according to the reference annotation are coreferent in the system output. The question that remains is *what* to extract as a recall error.

4.3.1 Entity Representation

In order to cope with the set-based nature of the task, as well as to be able to rely on a link-based notion of errors, we build our error analysis framework upon the directed graph representation of entities presented in Section 2.2. In particular, we employ the representation without dummy mentions. While dummy mentions are useful when devising approaches to coreference resolution, they are neither part of the reference annotation nor of the system output. Recall that in this entity representation, each entity is represented as a directed graph that is complete if we ignore the directions.

In the following, we mainly discuss how to extract recall errors. For extracting precision errors we switch the roles of reference and system entities.

Figures 4.1 and 4.2 show the graph representations of the BARACK OBAMA entity and of the system output. We augmented the system output with some spurious mentions

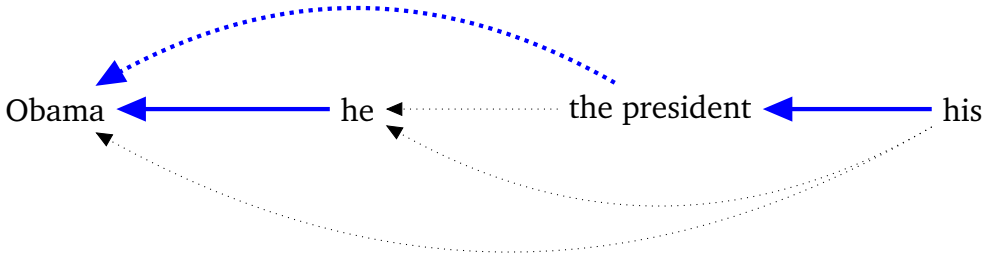

Figure 4.1: Graph representation G_d of the document d from Example (22). Dotted edges are candidates for errors with respect to the system output represented in Figure 4.2, subentities with respect to this output are the graphs containing {*Obama*, *he*} and {*the president*, *his*}. The blue edges constitute a spanning tree.

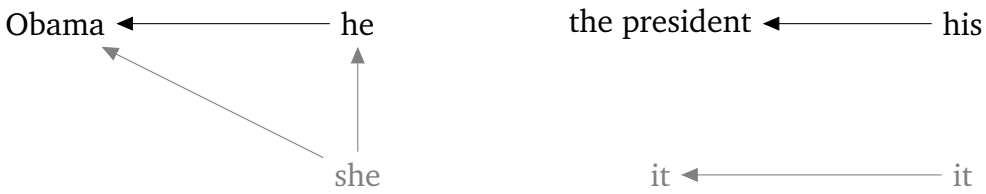

Figure 4.2: Graph representation G_d^S of an example system output for the document d from Example (22). We added some spurious mentions.

and links, in order to be able to show how our algorithm copes with this spurious information.

To obtain candidates for errors, we first consider all links from the representation G_d of the reference annotation which are not in the representation G_d^S of the system output. These edges are marked dotted in Figure 4.1. All of these links were missed by the system, since otherwise, via transitive closure, it would have identified the whole entity correctly.

Another perspective is that we are studying the partition P_d^S of the reference annotation G_d with respect to the system output graph G_d^S. This partition is the subgraph of G_d that contains only edges that are also in G_d^S. In the example displayed in Figure 4.1, the partition is the graph consisting of the solid edges. From this perspective, edges in G_d that are not in the partition are candidates for errors. We call the connected components of the subgraph *subentities*. These correspond to correctly resolved partial entities.

In our example, there are four candidates for errors: the links (*his*, *he*), (*his*, *Obama*), (*the president*, *he*) and (*the president*, *Obama*). In principle, we could extract all of these four links as errors, since resolving either of these links would result in a correctly identified entity. However, this inflates the number of errors. Note that it is sufficient to predict only *one* of these four links to obtain the correct entity by transitive closure. We therefore opt to choose a *representative* link from the list of candidate errors. If we choose a suitable criterion for representativeness, the errors extracted this way will give an accurate representation of the errors made by a system.

Which link to extract depends on the use case. Let us consider an example. Assume that the user prefers to extract errors with non-pronominal anaphors, which leaves the user with two candidates for errors: (*the president*, *he*) and (*the president*, *Obama*). In the candidate (*the president*, *he*), the mentions are close, but the error is composed of a non-pronominal anaphor and a pronoun antecedent. Such pairs are considered unreliable, some approaches even do not learn from or perform inference over these pairs (Ng and Cardie, 2002; Bengtson and Roth, 2008). The candidate (*the president*, *Obama*) is in some sense easier: since Barack Obama *is a* (US) president, information which is helpful in resolving the error can be mined from large corpora or found in knowledge bases such as Freebase (Bollacker et al., 2008). However, the distance between the mentions is larger than in the other error candidate. Hence, we can choose between two errors, and according to the task or the researcher's interest in the data, one error may be more suitable than the other. This corresponds to our third desideratum, flexibility in the notion of error.

4.3.2 Spanning Trees

Our aim now is to devise a method that extracts representative errors, where the measure of representativeness can be provided by the user. We propose a *spanning tree algorithm* for error extraction. This is motivated by the observation that it is sufficient for a coreference resolution approach to predict a *spanning tree* of the entity representation graph. The entity then can be inferred by transitive closure. Because of this, it is also sufficient to extract only one of the candidate errors displayed in Figure 4.1 as an error: if this error was resolved, we would have predicted the entity correctly. Therefore, to extract errors, we choose a spanning tree for each entity represented in G_d, and take all edges from the spanning tree that are not in the system output representation G_d^S as errors.

Before explaining the algorithm in more detail, we define the spanning tree of a directed graph.

Definition 12. *Let $G = (V, A)$ be a directed graph and let $u \in V$. A subgraph T is a spanning tree of G with root u if T is acyclic and every node other than u has out-degree 1, while u has out-degree 0.*

Since in our entity representation G_d edges point only in one direction, all subgraphs of G_d are acyclic. The spanning tree of each entity in G_d should be rooted in the first mention of the entity. Hence, if an entity in G_d contains the mentions $\{m_{i_1}, \ldots, m_{i_n}\}$ ($i_1 < \ldots < i_n$), the spanning tree should be rooted in m_{i_1}. For example, in Figure 4.1, the blue edges constitute a spanning tree of the entity representation graph. The blue dotted edge is a recall error: it is part of the spanning tree of the reference entity, but it cannot be found in the system output (Figure 4.2).

This suggests that we should construct spanning trees for each reference entity, and then extract those links as errors that do not appear in the system output.

Hence, we assume that we have algorithms T_{recall} and $T_{\text{precision}}$ for spanning tree construction at our disposal. T_{recall} takes two inputs: a connected component r of the reference annotation G_d, and the system output G_d^S. It then outputs a spanning tree of r. We also provide T_{recall} with the system output G_d^S in order to be able to devise spanning tree algorithm for reference entities that can take the system output into account. Analogously, $T_{\text{precision}}$ gets as input a connected component s of G_d^S and the reference annotation G_d. It outputs a spanning tree of s.

Algorithm 4.1 summarizes the whole method of error extraction.

4.4 Spanning Tree Variants

Our extraction algorithm is parametrized by spanning tree algorithms for reference and system entities, which are employed to extract recall and precision errors. Different spanning tree algorithms lead to different notions of an error. It is therefore crucial that we devise spanning tree algorithms which lead to useful notions of an error for coreference resolution researchers and system developers.

In this section, we present such algorithms. We discuss spanning tree algorithms for reference entities, which are used to extract recall errors, and spanning tree algorithms for system entities, which are used to extract precision errors. There are two options for presenting the algorithms. The first option is to devise weighting scheme

Algorithm 4.1. Error extraction from a corpus.

Input: A set of documents $\mathcal{D}$, a coreference resolution system S, spanning tree algorithms T_{recall} and $T_{\text{precision}}$

function EXTRACTERRORS($\mathcal{D}$, S, T_{recall}, $T_{\text{precision}}$)
  Set `recall_errors` $= [\,]$
  Set `precision_errors` $= [\,]$
  for $d \in \mathcal{D}$ **do**
    Let G_d be the representation of the reference annotation and G_d^S the representation of the system output
    for each connected component $r \in G_d$ **do**
      Compute a spanning tree $t_r = T_{\text{recall}}(r, G_d^S)$
      Add all edges in t_r not in G_d^S to `recall_errors`
    for each connected component $s \in G_d^S$ **do**
      Compute a spanning tree $t_s = T_{\text{precision}}(s, G_d)$
      Add all edges in t_s not in G_d to `precision_errors`

Output: Sets `recall_errors` and `precision_errors` of errors

for reference and system entity graphs based on notions of representativeness. We then can apply a standard algorithm for computing spanning trees such as Kruskal's algorithm (Kruskal, 1956) or Prim's algorithm (Prim, 1957). The second option is to devise an algorithm that chooses edges directly (based on representativeness). We opt for the second option, since choosing edges directly will facilitate the representation of the linguistic intuition behind the spanning tree algorithms. However, all algorithms we present can be transformed into weighting schemes, which then can serve as input to a standard spanning tree algorithm.

4.4.1 Spanning Tree Algorithms for Reference Entities

In most corpora in which coreference is annotated, the annotation consists of a mapping of mentions to entity identifiers: all mentions with the same entity identifier a coreferent. Only few corpora, such as MUC (Chinchor and Sundheim, 2003; Chinchor, 2001) also annotate links between mentions. Hence, in the general case, we have no information about the internal structure of the entity. Computing spanning trees of reference entities yields an internal structure. Intuitively, the spanning tree should consists of "easy" links: our approach missed those links, but, compared to other links, it would require the least effort to modify the system to correctly predict the link. Furthermore, the links should be interpretable and meaningful to humans for subsequent

analysis.

We are only interested in reference entity spanning trees such that the spanning tree restricted to each subentity also constitutes a spanning tree rooted in the first mention of the subentity[1].

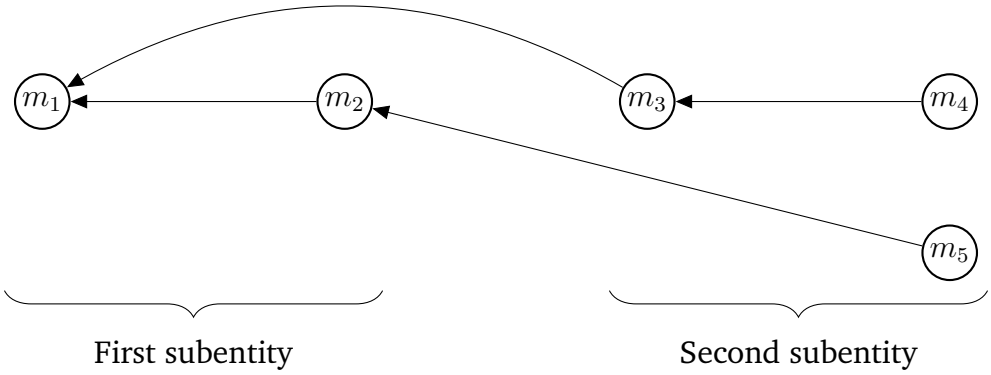

Figure 4.3: A spanning tree for a reference entity with two subentities. In our approach, this spanning tree is not valid for a reference entity.

The graph depicted in Figure 4.3 illustrates this issue. Suppose a reference entity consisting of the mentions $\{m_1, \ldots, m_5\}$ is partitioned into two subentities containing the mentions $\{m_1, m_2\}$ and $\{m_3, m_4, m_5\}$ respectively. If we take the spanning tree shown in Figure 4.3, we would extract two errors: (m_3, m_1) and (m_5, m_2). However, since $\{m_3, m_4, m_5\}$ is in the system output, m_5 was already resolved correctly to some antecedent, and we do not want to extract an error with m_5 as anaphor. To avoid extracting such an error, we first choose an arbitrary spanning tree for each subentity, rooted in the first mention of the subentity. Since the subentities also appear in the system output, all edges of each subentity's spanning tree are also in the system output. Therefore, we can choose any spanning tree for each subentity. We then choose the remaining edges of the spanning tree for the entity representation graph according to the spanning tree algorithm at hand.

In the following, we present three algorithms for spanning tree construction for reference entities. When provided with a reference entity r and the system output representation G_d^S, they first construct an arbitrary spanning tree for each subentity induced by G_d^S (rooted in the first mention of the subentity), for example by adding

[1] We enforce this requirement only for reference entity spanning trees, not for system entity spanning trees (see the discussion in Section 4.4.2).

an edge from each node in the subentity to the first mention of the subentity.

Choosing Spanning Trees by Distance. We first aim to devise a spanning tree algorithm that uses a criterion that is as simple as possible for extracting "easy" links. We opt to employ *mention distance* as a simple criterion for extracting "easy" links: the distance between two mentions can be computed without any information about their linguistic properties, and distance has successfully been used as a feature in many coreference resolution approaches, which induces that it can serve as a criterion for easiness.

Hence, if there are multiple candidate antecedents for an anaphor, we choose the closest one. Formally, we first construct arbitrary spanning trees for each subentity. Then, for each mention m_j ($j > 1$) which is the first mention of some subentity, we select the closest preceding mention m_i in the reference annotation. The edge (m_j, m_i) is then added as an edge of the spanning tree.

Choosing Spanning Trees by Accessibility. A closer look reveals that distance is mainly a good proxy for easiness for pronouns. Mention distance or variants of this distance have been successfully used as a main criterion in various pronoun resolution approaches (Hobbs, 1976; Lappin and Leass, 1994; Lee et al., 2013). In these approaches, distance is used as the main feature when deciding on an antecedent of a mention (after discarding antecedents which are incompatible, for example with respect to number, gender or semantic class).

However, using distance as a proxy for easiness is less helpful for common noun and proper name anaphors. Consider the following example taken from the OntoNotes 5.0 corpus[2] (Weischedel et al., 2013):

(23) [Investcorp, New York]$_1$, said [it]$_1$ and the management of [Sports & Recreation Inc.]$_2$ bought the operator of [the 10-store Sports Unlimited chain]$_3$ for some $40 million.

[The investment bank]$_1$ becomes majority shareholder in [Sports & Recreation, a 10-year-old sporting goods retailer]$_2$, said Oliver E. Richardson, a member of [Investcorp's]$_1$ management committee and a director of [the chain]$_3$.

[2] Document `wsj_2422`.

Let us consider the reference entity INVESTCORP, which is referred to by the mentions *Investcorp, New York*, *it*, *the investment bank* and *Investcorp's*.

One of our best-performing systems which we will present in Chapters 6 and 8, based on a mention ranking architecture, does not recognize the whole entity, but outputs {*Investcorp*, *it*} and does not assign *the investment bank* and *Investcorp's* to any coreference chain. Applying the error extraction algorithm based on spanning trees defined by distance, we would extract the two errors (*the investment bank*, *it*) and (*Investcorp's*, *the investment bank*).

The errors are not very helpful: (*the investment bank*, *it*) consists of a non-pronominal anaphor and a pronoun antecedent. This type of link is considered unreliable (see the discussion in Section 4.3.1). In (*Investcorp's*, *the investment bank*) the mention are in an ISA relation, but it would be more helpful to provide the user with the error (*Investcorp's*, *Investcorp, New York*), since based on that error, the user can just analyze and improve the alias feature in his or her coreference resolution system. Hence, we need a more meaningful definition of an *easy* link than just according to distance. Our brief analysis showed that mention types and (partial) string matches provide clues for easiness.

We devise a criterion for easiness inspired by *accessibility theory* (Ariel, 1988), a theory that links choice of referring expressions to the mental representation of the entities these expressions are used to refer to. According to accessibility theory, referring expressions can be broadly divided into low, mid and high accessibility markers. More accessible entities are referred to by higher accessibility markers. In particular, "Low Accessibility marked entities are often those stored in long-term memory, while High Accessibility marked entities are normally those held in short-term memory" (Ariel, 1988, p. 80). Proper names and definite descriptions are low accessibility markers, while pronouns are high accessibility markers. Regarding the relation between referring expressions and their antecedent, Ariel states that "most High Accessibility markers refer to unmarked, contextually salient entities (especially discourse topics)" and that "Low Accessibility markers refer to more marked, less accessible antecedents" (Ariel, 1988, p. 82f.). Accessibility theory suggests a criterion for easiness based on this relation: if a high accessibility marker is used, choose a very accessible antecedent (such as the closest one). If a low accessibility marker is used, choose an antecedent which itself is expressed by a low accessibility marker.

Let us return to our INVESTCORP example (Example (23)). Figure 4.4 visualizes all candidate errors as non-solid edges. Consider the anaphor *the investment bank*. Ac-

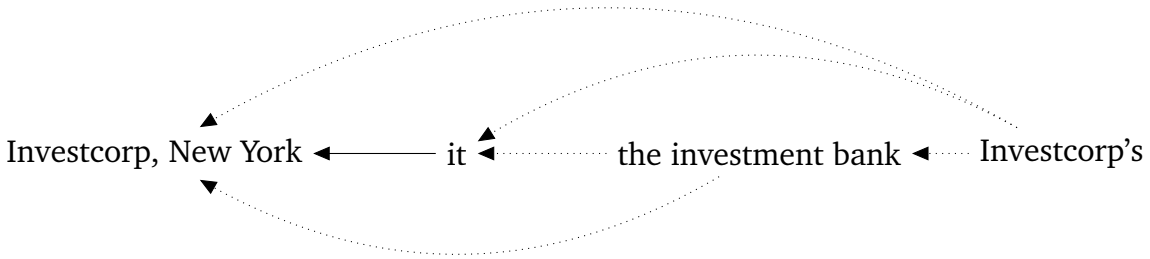

Figure 4.4: Visualization of candidate errors for the INVESTCORP example.

cording to the spanning tree algorithm based on distance, (*the investment bank*, *it*) would be extracted as an error. According to accessibility theory, the link (*the investment bank*, *Investcorp, New York*) is much easier: *the investment bank* is a definite description, and therefore a low accessibility marker. It refers to a less accessible antecedent, such as *Investcorp, New York*. This also is in line with our intuition: (*the investment bank*, *it*) is unreliable, since *it* is potentially a candidate antecedent for many different anaphors. (*the investment bank*, *Investcorp, New York*), however, constitutes a well-defined ISA relation, which can be queried from knowledge bases or large corpora.

Algorithm 4.2 presents our spanning tree algorithm based on accessibility theory. It operationalizes the criterion that low accessibility markers necessitate antecedents expressed by low accessibility markers, and that high accessibility markers are used when the antecedent is very accessible. Given a reference entity r and the system output G_d^S, we first construct arbitrary spanning trees for the subentities induced by G_d^S, rooted in the first mention of the subentities. We choose the remaining edges of the spanning tree based on a notion inspired by accessibility theory. To do so, we iterate through all mentions which have an out-degree of 0 in the partially constructed spanning tree. For pronouns – high accessibility markers – we choose the closest preceding mention in r and add this as an edge. For the remaining mentions, which are all low accessibility markers, we first look for preceding mentions with string or head match. If we could not find such mentions, we choose the antecedent which is marked with the lowest accessibility (since the low accessibility marked anaphor refers to a less accessible antecedent).

For our example, this algorithm chooses edges such that the errors (*the investment bank*, *Investcorp, New York*) and (*Investcorp's*, *Investcorp, New York*) are extracted.

Algorithm 4.2. Spanning tree construction based on accessibility.

Input: Graph representation r of a reference entity, graph representation G_d^S of system output

function ACCESSIBILITYSPANNINGTREE(r, G_d^S)
  Let t be an empty graph
  for each subentity r_i of r induced by G_d^S **do**
    Construct a spanning tree t_i for r_i, rooted in the first mention of r_i
    Add the edges of t_i to t
  for each mention m_j with out-degree 0 in t **do**
    if m_j is a pronoun **then**
      Add (m_j, m_i) to t, where m_i is the closest preceding mention in r
    else if there is a preceding mention in r with a full string match **then**
      Add (m_j, m_i) to t, where m_i is the closest such mention
    else if there is a preceding mention in r with a head string match **then**
      Add (m_j, m_i) to t, where m_i is the closest such mention
    else if there is a preceding proper name mention in r **then**
      Add (m_j, m_i) to t, where m_i is the closest such mention
    else if there is a preceding common noun mention in r **then**
      Add (m_j, m_i) to t, where m_i is the closest such mention
    else
      Add (m_j, m_i) to t, where m_i is the closest preceding mention in r

Output: A spanning tree t of r

Choosing Spanning Trees by Pairwise Scores. The spanning tree algorithms we have considered so far were based on linguistic intuition. We used this linguistic intuition to devise heuristics for extracting spanning trees.

An alternative is to consider a *data-driven* approach: can we *learn* from training data what constitutes an easy link? Let us revisit our definition of easiness from the beginning of this section: a link is easy if, compared to other candidate links, it would require the least effort to modify our system to correctly predict the link.

As we saw in Chapter 3, most coreference resolution approaches are based on scoring links between mentions. They typically assign a score $s(m_j, m_i)$ to each pair (m_j, m_i) of mentions. This induces a ranked list of candidate antecedents for each anaphor m_j: for two candidate antecedents m_i and m_k we have $m_i >_{\text{score}} m_k$ if $s(m_j, m_i) > s(m_j, m_k)$. This suggests a spanning tree algorithm for reference entities: first construct arbitrary spanning trees rooted in the first mention for the subentities. Then, for each mention m_j with out-degree 0, choose the edge (m_j, m_i) such that $s(m_j, m_i) > s(m_j, m_k)$ for all preceding $m_k \neq m_i$. Hence, for each mention m_j which is first in its subentity, the edge (m_j, m_i) in the spanning tree received a higher score than the alternatives for the same anaphor. If we interpret the pairwise scores as confidence values, the system deems it more likely that the pair (m_j, m_i) is coreferent, compared to the alternative antecedents from the reference entity.

4.4.2 Spanning Tree Algorithms for System Entities

We now consider spanning tree algorithms for system entities. Intuitively, the spanning tree for a system entity should consist of the most reliable links between mentions in the entity. Let us call such links "reliable".

When computing spanning trees for reference entities, we did not allow outgoing edges from mentions that are not first in their respective subentity (see Figure 4.3). The rationale was that subentities of reference entities correspond to correctly resolved partial entities, therefore mentions that are not the first mention in their subentity have already been resolved to a correct antecedent. For system entities, subentities correspond to parts of a system entity that contain only coreferent mentions. Any mention in such a subentity may be responsible that a wrong link was induced which merged two subentities. Therefore, we do not put any restrictions on spanning tree construction for system entities.

Typically, coreference resolution systems do not just output the mapping of men-

tions to entities, but do also output the structure of the found coreference chains, which we can harness for spanning tree constructions. Hence, for system entities, we consider two classes of spanning tree algorithms: the first class deals with the general case, where no additional structure is available. The second class takes into account additional structural information provided by the system output.

Applying a Reference Entity Spanning Tree Algorithm. When computing spanning trees for reference entities, we are interested in "easy" links. For system entities, we are interested in "reliable" links.

We can apply the algorithms for extracting "easy" links to construct spanning trees for system entities. Hence, this leads to criteria for importance based on distance, accessibility and pairwise scores. We have designed the heuristic-based spanning tree algorithms to yield links meaningful to and interpretable by humans. Based on the assumption that distance or accessibility are crucial factors in anaphoric reference, the system entity spanning trees obtained by these algorithms consist of "reliable" links. The data-driven approach based on pairwise scores also can be used to construct spanning trees consisting of "reliable" links: if a link receives a high score by the system, it is deemed a reliable link by the system.

Regarding the differences between the individual approaches, the observations from the previous section on reference entity spanning trees apply.

Choosing Spanning Trees Based on Pairwise Output. As we have already discussed, the vast majority of approaches to coreference resolution are based on scoring pairwise links between mentions. The approaches then consolidate these pairwise scores to construct coreference chains.

For many approaches such as mention pair with best- or closest-first clustering (Soon et al., 2001; Ng and Cardie, 2002), mention ranking (Denis and Baldridge, 2008) and antecedent trees (Fernandes et al., 2014), the links which are chosen in the consolidation step already constitute spanning trees of system entities. This holds even for some mention-entity models: each anaphor gets assigned to at most one preceding cluster, and the decision is often guided by the relationship of the anaphor to one particular mention in the preceding cluster (e.g. Webster and Curran, 2014).

Hence, for approaches that already output spanning trees of the system entities, we can just take these spanning trees for error extraction. For approaches that output a set of pairwise links that do not constitute a spanning tree (for example by listing

all anaphor-antecedent pairs recognized by the system), we need to switch back to the adapted reference entity spanning tree algorithms, which are applied to the graph induced by the set of links in the output. For the approaches which do not even output links, we also switch back to the adapted algorithms, as described above.

4.5 Relation to Evaluation Metrics

The analysis framework we just presented extracts errors which are then subject to further (human) analysis. Coreference resolution evaluation metrics, on the other hand, quantify the errors made by a system, and use the errors to derive the precision and recall of the system. In this section, we investigate the relationship of our analysis framework to evaluation metrics in detail.

Following Chen and Ng (2013) we distinguish between *linguistically agnostic metrics* and *linguistically aware metrics*. Linguistically agnostic metrics do not employ linguistic information, while linguistically aware metrics take linguistic information into account during scoring.

4.5.1 Linguistically Agnostic Metrics

All the evaluation metrics discussed in Section 2.4 (MUC, B^3, $CEAF_e$, $CEAF_m$ and BLANC) are linguistically agnostic, since they do not differ between different mention or entity types when evaluating. B^3 and the variants of CEAF are not founded on a link-based structure, but take a set-based perspective. Hence, they do not provide means to extract link-based errors. We leave determining whether the framework of these metrics exhibits a useful notion of errors to future work. Like our framework, MUC and BLANC are link-based. We now discuss differences and similarities of the proposed error analysis method compared to MUC and BLANC.

4.5.1.1 MUC

Our framework is based on the same entity representation as the MUC metric. Recall the definition of MUC (Section 2.4.2): for computing recall, iterate through every connected component (i.e. entity) g in the reference entity representation graph[3]. For each entity, partition the graph with respect to the system output. Then compute

[3]For computing precision, switch the roles of reference and system entities.

any spanning tree t of g with the following property: t restricted to each connected component in the partition is also a spanning tree of the connected component. Then *recall error* re for g is the fraction of edges of t that are not in the partition. *Recall* is $1 - re$. To extend recall error and recall to the whole document, the sum is computed over all spanning trees before computing the fraction.

Now compare this algorithm with our error extraction algorithm based on spanning trees: the only difference is that we compute a *particular* spanning tree of g, while MUC just takes an arbitrary spanning tree (both constrained such that they are also spanning trees for the connected components of the partition). Since MUC aims to quantify errors, while the analysis framework aims to extract errors for qualitative analysis, also the output differs. However, both rely on the edges $E = \{e_1, \ldots, e_m\}$ which are in the spanning tree but not in the partition. The analysis framework outputs E directly, while MUC condenses this information into $1 - |E| / |t|$.

Hence, our analysis framework can be understood as an adaption of the MUC metric for error extraction. We achieve this by replacing the generic spanning tree algorithm with tailored spanning tree algorithms, which lead to different notions of errors.

Several shortcomings of the MUC score have been identified by Bagga and Baldwin (1998): First, the MUC score does not take the correct identification of singleton mentions into account[4]. These lead to entities consisting of a single mention, which cannot be represented by spanning trees. Second, the score does not consider the size of subentities when scoring coreference chains. To see why this may lead to unintuitive results, consider the following example, which is a slight variation of the example discussed by Bagga and Baldwin (1998). Assume that a document contains three reference entities, $e_1 = \{m_1, m_2, m_3, m_4\}$, $e_2 = \{m_5, m_6\}$ and $e_3 = \{m_7, m_8, m_9, m_{10}\}$. One system S_A puts e_1 and e_2 into the same system entity, another system S_B puts e_1 and e_3 into the same system entity. Bagga and Baldwin (1998) argue that S_A should obtain a higher score that S_B, since it conflated a large entity and a small entity, while S_B conflated two large entities. However, since the MUC score only considers spanning trees, both outputs receive the same score: a recall of 1 and a precision of 7/8.

Both of these issues concern *scoring* and are not valid for our error extraction algorithm: First, if the corpus is annotated with singleton mentions, we still extract all errors involving singleton mentions, since the disability to reward identification of singleton mentions does not affect error extraction. Second, we aim to extract errors, and

[4] Singleton mentions are mentions that refer to some entity, but do not corefer with any other mention in the document.

therefore do not encode precision/recall of a system output into a single number. The system outputs from the examples above lead to different errors. If the user wishes to favor severe errors for extraction, the user can devise a new spanning tree algorithm which takes the size of the subentities into account.

4.5.1.2 BLANC

As our framework, BLANC is link-based. In particular, BLANC considers *all* links between coreferent mentions, and between non-coreferent mentions. These sets are constructed for the reference annotation and for the annotation induced by system output. This enables a classification of links into true positives, false positives, true negatives and false negatives. Based on this, recall, precision and F_1 score for coreferent mentions and non-coreferent mentions are computed, and then averaged.

The main difference between the scoring method of BLANC and the extraction method of our framework is that BLANC does not rely on spanning trees to represent entities, but chooses all links between mentions to represent entities (similar to our graph-based entity representation detailed in Section 2.2). Since BLANC relies on an error classification in terms of true/false positives and negatives, this suggests that this classification can be used for error extraction. False positives are precision errors, while false negatives are recall errors.

However, we argue that errors extracted this way are not useful. If for example a reference entity is split into two by a system, all inter-entity links between the mentions in the subentities would be extracted as recall errors. This results in a large number of errors, which are difficult to process for the user. We therefore believe that adapting BLANC's scoring method does not result in a useful error extraction algorithm.

4.5.2 Linguistically Aware Metrics

Recent work on evaluation metrics takes linguistic information into account. Chen and Ng (2013) devise a unified graph-based representation for different existing evaluation metrics, and inject linguistic knowledge by weighting specific links in this graph. Tuggener (2014) proposes new metrics, which are tailored for particular applications such as summarization. To do so, he redefines the notion of a correct link depending on the task in focus. Both of these works only consider scoring, but weight or distinguish links in the reference and system entities, which in principle allows for error extraction. However, the authors do not attempt to extract errors, and it is not

clear whether any errors extracted that way could be useful for analysis and system development.

5 A Machine Learning Framework for Coreference Resolution

In this chapter we present a machine learning framework for coreference resolution. Our framework is based on the observation that many machine learning approaches to coreference resolution can be characterized by the *latent structures* they operate on. Hence, we develop a machine learning framework that explicitly models these latent structures, building upon the graph-based representation for coreference which we devised in Section 2.2.

We first revisit our discussion of coreference resolution approaches from Chapter 3 and note that we can understand the approaches as predicting structures not observed in the data (Section 5.1). Motivated by this insight, we describe a general machine learning setting for structured prediction with latent variables (Section 5.2). Next, we tailor this setting to the special case of coreference resolution (Section 5.3). Our framework can handle arbitrarily complex structures. Complex structures may necessitate complex or approximate inference methods. We therefore discuss how such methods are incorporated in the framework (Section 5.4). To train models in the framework, we present a structured perceptron with latent variables and cost-augmented inference (Section 5.5).

5.1 Underlying Structures

In Section 3.1, we discussed machine learning approaches for coreference resolution, ranging from simple mention pair models to sophisticated entity-centric approaches. In our discussion, we characterized each approach across different dimensions: the structure the approach operates on, the machine learning method used, the feature set used, and how the approach selects and handles training data.

From this perspective, the structure that underlies the approach is most defining for

the approach: Researchers devise more sophisticated structures to obtain a more adequate or better-performing model for coreference resolution. The remaining dimensions are chosen in concordance: the machine learning classifier should be suitable for the structure; the features make use of the structures; and the selection and handling of training data is typically optimized to give optimal performance with respect to the new structure obtained (Ng and Cardie, 2002).

Hence, since the underlying structures are most defining for approaches to coreference resolution, let us have a closer look at the role of these structures for predicting coreference relations.

From a formal perspective, the task of coreference resolution is to predict the equivalence classes of the coreference relation in one document. In particular, the expected output of a coreference resolution approach is a segmentation of the mentions into equivalence classes. This segmentation can be represented by a mapping of mentions to entity identifiers, or by graphs as described in Section 2.2. However, as we have observed in Section 2.3, this output is too complex to be predicted directly. Approaches have to resort to simplifications or approximations, from which the final output is derived in a postprocessing step. The different structures employed by different approaches correspond to different ways of simplifying or approximating the prediction problem of equivalence classes.

To understand this better, we discuss three structures in detail: mention pair models (Soon et al., 2001), mention ranking models (Denis and Baldridge, 2008) and mention-entity models (Yang et al., 2008). Mention pair models approximate the prediction by performing binary classification of mention pairs. They first create a list of mention pairs, and then handle each pair in isolation and label it as coreferent or non-coreferent. Ranking models, on the other hand, construct pairs consisting of the anaphor in focus m_j and the list of all candidate antecedents, $\{m_0, \ldots, m_{j-1}\}$. The approach then chooses the antecedent of m_j from the list. Similarly, mention-entity models rely on pairs consisting of the anaphor m_j and the list of partially constructed entities so far, say $\{\{m_0\}, \{m_1, m_3\}, \{m_4\}, \{m_2, m_5, \ldots, m_{j-1}\}\}$. They decide to which partial entity m_j should be added, or whether it should start a new entity.

These structures have in common that they are *not annotated in the data*. To be able to cope with the complex prediction problem for coreference resolution, the approaches introduce auxiliary structures, from which the equivalence classes are extracted during postprocessing. Hence, in terms of machine learning, we can understand approaches to coreference resolution as performing *latent structured prediction*.

This yields a unified perspective on machine learning approaches to coreference resolution. Such a perspective enables us to highlight structural differences and similarities, and also establishes a setting to systematically analyze and compare approaches in terms of the structures they employ. Furthermore, we can define new approaches and transfer design decisions between different approaches.

In the remainder of this chapter, motivated by the observations above, we present a machine learning framework that models coreference resolution as prediction of latent structures. Our framework generalizes previous work on latent antecedents and trees for coreference resolution (Yu and Joachims, 2009; Chang et al., 2012; Fernandes et al., 2014). We use the mention ranking model with latent antecedents (Chang et al., 2012) as a running example.

5.2 General Setting

Formally, coreference resolution is a *prediction task*: Given a document, we want to predict the coreference chains over mentions in that document. From a machine learning perspective, the goal is to learn a function

$$f\colon \mathcal{X} \to \mathcal{Y}, \tag{5.1}$$

where $\mathcal{X}$ is the *input space* and $\mathcal{Y}$ is the *output space*. In our case, $\mathcal{X}$ consists of documents and $\mathcal{Y}$ consists of the documents enriched with coreference annotations. We have seen that coreference resolution approaches actually predict latent structures from which the coreference chains are induced. Hence $\mathcal{Y}$ factors into a *latent output space* $\mathcal{H}$ and an *observed output space* $\mathcal{Z}$:

$$f\colon \mathcal{X} \to \mathcal{H} \times \mathcal{Z}. \tag{5.2}$$

For coreference resolution, $\mathcal{H}$ contains the latent structures which are used to infer coreference relations, while $\mathcal{Z}$ contains the coreference relations. Given $x \in \mathcal{X}$, we write $\mathcal{H}_x$ and $\mathcal{Z}_x$ for the output spaces only encoding structures built upon x. For instance, in the mention ranking model with latent antecedents (Chang et al., 2012), $\mathcal{H}_x$ contains all possible antecedent decisions for each mention in a document $x \in \mathcal{X}$.

To model the prediction, we assume that there is a function

$$F\colon\ \mathcal{X} \times \mathcal{H} \times \mathcal{Z} \to \mathbb{R} \tag{5.3}$$

that scores triples consisting of an input, the latent output and the observed output. For an input $x \in \mathcal{X}$, we obtain the prediction by computing the maximum scoring output, i.e. we define

$$f(x) = \operatorname*{arg\,max}_{(h,z)\in\mathcal{H}_x\times\mathcal{Z}_x} F(x,h,z). \tag{5.4}$$

We consider only *linear models*. Such models represent a triple $(x,h,z) \in \mathcal{X} \times \mathcal{H}_x \times \mathcal{Z}_x$ using a *feature function*

$$\phi\colon\ \mathcal{X} \times \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^d \tag{5.5}$$

for some $d \in \mathbb{N}$. ϕ projected to one particular dimension is called a *feature*. ϕ is used to jointly describe the input, the output and the latent structure. For example, many approaches – such as the ranking model – rely on evaluating candidate anaphor-antecedent pairs. With the feature function ϕ, we can describe the linguistic properties of anaphor and antecedent, and the relation between the mentions. Typical features are the mention types of anaphor and antecedent, or whether there is a string match[1].

To score (x,h,z), we compute the scalar product of the feature representation with some parameter vector $\theta \in \mathbb{R}^d$, which is learned from data (see Section 5.5). That is, the scoring function is

$$F(x,h,z) = \langle \theta, \phi(x,h,z) \rangle, \tag{5.6}$$

and therefore

$$f(x) = \operatorname*{arg\,max}_{(h,z)\in\mathcal{H}_x\times\mathcal{Z}_x} \langle \theta, \phi(x,h,z) \rangle. \tag{5.7}$$

We write f_θ instead of f to emphasize the parametrization of the prediction function with respect to $\theta \in \mathbb{R}^d$.

5.3 Modeling Coreference Resolution

Any instantiation of the generic framework discussed above needs to define the input space $\mathcal{X}$, the latent output space $\mathcal{H}$ and the observed output space $\mathcal{Z}$. In this section, we introduce an instantiation that employs graphs to express the latent structures.

[1] To convert these categorical and boolean features into real-valued features, we add one binary feature (with value either 0 or 1) for each value of the feature.

Under mild constraints, which we discuss, this representation allows for efficient inference to solve Equation 5.7.

5.3.1 The Input Space $\mathcal{X}$

The task of coreference resolution is to map documents to documents annotated with coreference chains. Hence, $\mathcal{X}$ contains representations of documents. We do not make this representation explicit. Instead, we make the simplifying assumption that we have access to all linguistic information which may be needed for feature extraction. In particular, we assume that system mentions were already extracted. Using the notation established in Chapter 2, we write $M_x = \{m_1, \ldots, m_n\}$ for the system mentions in $x \in \mathcal{X}$[2].

One issue for automatic coreference resolution is *anaphoricity determination* (see for instance Ng (2004)). A mention $m \in M_x$ either does have an antecedent or is not anaphoric. In order to model anaphoricity determination, we introduce a *dummy mention* m_0 and write $M_x^0 = \{m_0\} \cup M_x$. If m_0 is determined to be the antecedent of a mention m, this is equivalent to making the prediction that m does not have any antecedent (see also Section 2.2).

5.3.2 The Latent Output Space $\mathcal{H}$

Given a document $x \in \mathcal{X}$, the latent output space $\mathcal{H}_x$ includes all structures over the mentions in x that guide the approach to predict the coreference chains. Since different approaches employ different latent structures, the latent output space differs per approach. As we have developed a graph-based view on coreference in Chapter 2.2, which was also the basis of our error analysis framework, we will employ a graph-based representation of these latent structures as well.

As a starting point, we consider *labeled subgraphs* of the graph representation with dummy mentions. Recall that in these graphs, each equivalence class of mentions corresponds to a fully connected component, adhering to directionality constraints. The first mention in each component is connected to the dummy mention m_0. The subgraphs model pairwise relations between mentions, have a notion of directionality, which is useful for expressing anaphor-antecedent relations, and can incorporate additional information via edge labels.

[2]We describe our algorithm for mention detection in detail in Section 8.1.2

However, such graphs are not expressive enough for our purposes. As we saw in discussion of entity-centric models, some of these approaches build coreference chains by predicting that sets of mentions, say $\{m_1, m_3, m_4\}$ and $\{m_2, m_5\}$, are coreferent. Hence, we need a formalism that is expressive enough to model such relations. We employ a generalization of directed graphs, *directed hypergraphs*.

Definition 13. *A directed* hypergraph H *is a tuple* $H = (V, A)$ *where* V *is some set of nodes and*

$$A \subseteq \{(X, Y) \mid X, Y \subseteq V, X \cap Y = \emptyset\}. \tag{5.8}$$

Hyperedges $a \in A$ consist of two (possibly empty) sets, a *tail* X and a *head* Y. Directed graphs are a special case of directed hypergraphs: A directed graph is a directed hypergraph where the tail and head of each edge have cardinality one.

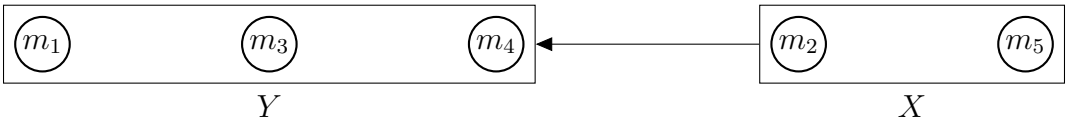

Figure 5.1: A hyperedge (X, Y) that signals that two sets of mentions are coreferent. $X = \{m_2, m_5\}$ is the tail of the hyperedge, $Y = \{m_1, m_3, m_4\}$ is the head of the hyperedge.

Since hypergraphs can model relations between sets of mentions, they are expressive enough to also account for approaches that predict that sets of mentions are coreferent. Figure 5.1 shows an hyperedge that connects two sets of mentions, modeling that these sets are coreferent.

Directed hypergraphs serve as our representation for latent structures. Formally, a *valid latent structure* for a document $x \in \mathcal{X}$ is any (labeled) directed hypergraph $G = (V, A, L_A)$ with the following properties:

- the set of nodes are the mentions (including the dummy mention), $V = M_x^0$,
- the edges are a subset of all hyperedges where the dummy mention cannot appear in the tail,

$$A \subseteq \left\{(X, Y) \mid X \subseteq M_x, Y \subseteq M_x^0, X \cap Y = \emptyset\right\}. \tag{5.9}$$

- $L_A \colon A \to L$ is a function that assigns to each edge $a \in A$ a label from a set of labels L.

When considering the case of directed graphs, the edges correspond to anaphor-antecedent decisions. Otherwise, they correspond to coreference relations between sets of mentions.

We denote the set of all directed hypergraphs that adhere to the definition above as $\mathcal{G}$. Depending on the approach in focus, we will consider different classes of labeled directed (hyper)graphs. For the mention ranking model, for example, the latent structure encodes the antecedent decision for each anaphor. Each mention should have exactly one antecedent (where the dummy mention is a valid antecedent). Therefore we consider all graphs $h = G = (V, A, L_A) \in \mathcal{G}$ which satisfy (i) for all $m_j \in V$ $(j > 0)$, there exists exactly one $i < j$ such that $(m_j, m_i) \in A$ and (ii) $L_A(a) = \texttt{None}$ for each $a \in A$ (the mention ranking model does not employ any labels). The set of graphs $h \in \mathcal{G}$ that satisfy these two requirements constitute the set of latent structures $\mathcal{H}$ for the mention ranking approach.

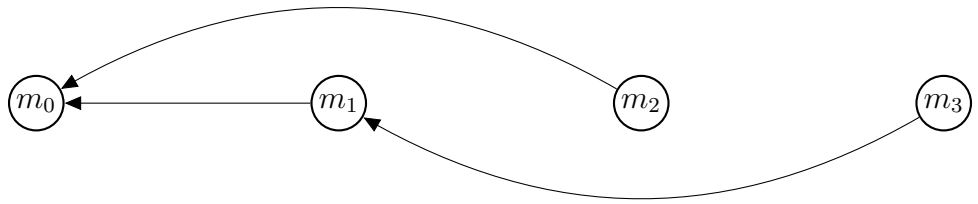

Figure 5.2: An example latent structure for the mention ranking approach, encoding a particular set of anaphor-antecedent decisions. All edges have the `None` label, which is not shown in this figure.

Figure 5.2 shows an example graph. In Chapter 6 we will consider many more structures and we will also see how edge labels are useful for modeling approaches.

5.3.3 The Observed Output Space $\mathcal{Z}$

While the latent structures encode the coreference relations between mentions, they are not in the format to match with the annotations in the corpora. In these annotations, each mention is mapped to an *entity identifier*. We model this mapping via functions from mentions to integers. Let $x \in \mathcal{X}$ be some document. The observed output space $\mathcal{Z}_x$ for x consists of all functions

$$z\colon\ M_x \to \mathbb{N}. \tag{5.10}$$

Two mentions $m_i, m_j \in M_x$ are considered coreferent if and only if $z(m_i) = z(m_j)$. Hence, $\mathcal{Z}_x$ represents all possible coreference chains over the mentions in x.[3]

5.3.4 Relating $\mathcal{H}$ and $\mathcal{Z}$

Typically, we obtain the output z by aggregating the decisions encoded in the predicted latent structure h. For example, in the mention ranking model, we take the transitive closure over all antecedent decisions expressed in the latent structure, ignoring the dummy mention m_0 (since these edges do not signal coreference, but non-anaphoricity). In the example shown in Figure 5.2, this procedure outputs the entities $\{m_1, m_3\}$ and $\{m_2\}$.

Formally, this can be expressed via a function

$$\texttt{obtain\_coreference}\colon \mathcal{H} \to \mathcal{Z}$$

that maps the latent structures to the corresponding coreference information. If z and h encode the same coreference information, that is if $z = \texttt{obtain\_coreference}(h)$, we call h and z *consistent*[4]. When listing pairs of latent structures and coreference information, as in $(h, z) \in \mathcal{H} \times \mathcal{Z}$, we slightly abuse notation to avoid complicated terminology and always assume that h and z are consistent.

5.3.5 Feature Factorization

In order to make this framework usable, we must take care that the maximization problem to obtain the best-scoring structures in Equation 5.7 is feasible. However, for many structures, the size of the search space $\mathcal{H}_x$ is exponential in the number of mentions $|M_x|$ appearing in the document x.

To efficiently find the maximum scoring structure, many approaches are based only on directed graphs, not hypergraphs, and assume that the feature function ϕ factors with respect to the edges of the directed graph. As we will see, the combination of such a factorization with directed graph structures makes the maximization problem

[3] Since the correctness does not depend on the integer values of the entity identifiers, we consider two functions $z, z' \in \mathcal{Z}_x$ as equal if it holds that $z(m_i) = z(m_j)$ if and only if $z'(m_i) = z'(m_j)$.

[4] We will often consider substructures h' of h that only encode coreference information for a subset of all mentions in the input document. We call z and h' consistent if z and $\texttt{obtain\_coreference}(h')$ agree on this subset of mentions.

feasible. Furthermore, since the latent structure already encodes all coreferential information, these approaches also assume that the feature function is independent from z.

Formally, we have

$$\phi(x, h, z) = \sum_{a \in A} \phi(x, a) \tag{5.11}$$

for $(x, h, z) \in \mathcal{X} \times \mathcal{H}_x \times \mathcal{Z}_x$ and $h = (V, A, L_A)$.

Entity-based models do not adhere to this feature factorization: The features employed by these models access richer information about the latent structures h.

5.3.6 Substructures

The graph representation for latent structures which we introduced represents the coreference information on the *document level*: The graph encodes the coreference decisions for the whole document. However, some approaches split the prediction into several subproblems for each document. For instance, the mention ranking model considers each anaphor in isolation, and the mention pair model considers the coreference decision between each pair as an individual problem.

To account for this in our framework, we introduce the notion of *substructures*. To define these, we assume that each approach we consider also has a *substructure-inducing function*

$$\texttt{sub}\colon \mathcal{X} \to \{N \mid N \subseteq 2^{\mathcal{G}}\} \tag{5.12}$$

that assigns to each document $x \in \mathcal{X}$ a set of *substructure spaces* $\mathcal{H}_{x,1}, \ldots, \mathcal{H}_{x,m}$. Typically, $h_i \in \mathcal{H}_{x,i}$ is a latent structure $h \in \mathcal{H}_x$ restricted to a subset of the mentions appearing in x.

With these substructures, the maximization problem in Equation 5.7 factors into m subproblems (as defined by the number of substructure spaces). The ith problem is

$$f_i(x) = \operatorname*{arg\,max}_{(h_i, z) \in \mathcal{H}_{x,i} \times \mathcal{Z}_x} \langle \theta, \phi(x, h_i, z) \rangle. \tag{5.13}$$

To obtain a structure $\hat{h} \in \mathcal{H}_x$ from the predicted substructures $\hat{h}_1, \ldots, \hat{h}_m$, where $\hat{h}_i = (V_i, A_i, L_{A_i})$, we take the union over all substructures, that is we set $V = \cup_{i=1}^m V_i$, $A = \cup_{i=1}^m A_i$ and $L_A(a) = L_{A_i}(a)$ if $a \in A_i$. We assume that sub is designed such that there are no contradictions in the output of the individual substructure predictions.

Figure 5.3 displays substructures for the mention ranking problem. The antecedent

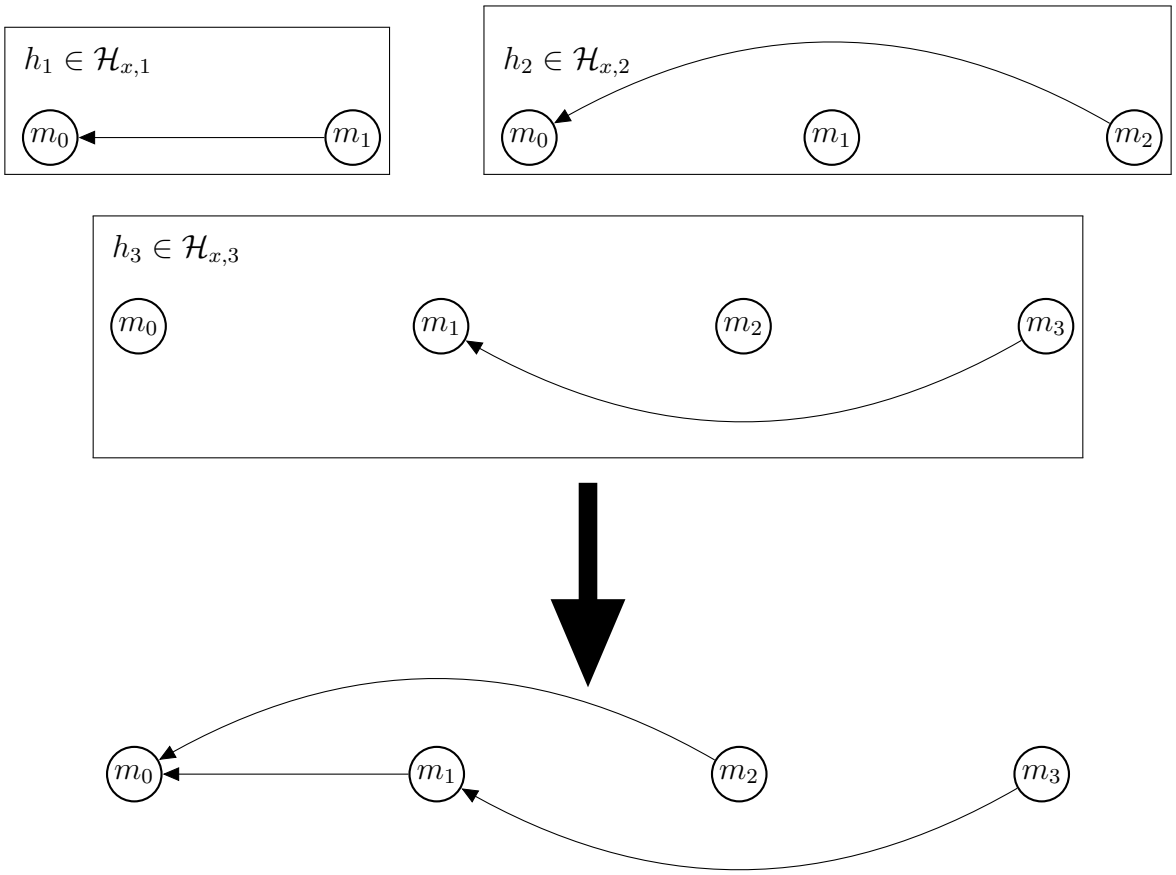

Figure 5.3: Substructures for the mention ranking approach.

decision for each anaphor is modeled as an individual problem, which corresponds to substructures h_1, h_2 and h_3. To obtain the latent structure h for the whole document, we take the union over all nodes and edges.

5.4 Inference

In this section we consider the maximization problems described by Equations 5.7 and 5.13 in more detail. To solve these maximization problems, we need to find the overall highest-scoring structure among a set of structures (according to a linear model). The set may be too large to explicitly enumerate all structures. Therefore, it is not trivial to obtain a solution to Equations 5.7 and 5.13.

We now discuss various classes of inference algorithms in general. We discuss specific algorithms for the latent structures considered in this thesis in Chapter 6.

5.4.1 Greedy Inference

For a large class of structures including the mention-pair model (Soon et al., 2001), mention ranking (Durrett et al., 2013) and antecedent trees (Fernandes et al., 2014), the maximization problems can be solved by *greedy inference*: We break the maximization problems down into individual, independent steps and make a locally optimal decision at each step. By design, aggregating all locally optimal decisions will lead to an overall optimal decision.

5.4.2 Incremental Inference

Not all approaches are based on latent structures where optimal solutions can be computed by greedy inference. Consider, for example, entity-based models. These compute features over sets of mentions (also called *partial entities*) that are assumed to be coreferent. Hence, when deciding whether to attach a mention m_j to a partial entity $\{m_{i_1}, \ldots, m_{i_k}\}$, this decision depends on all decisions for mentions that are already in some partial entity. Therefore, decisions are not independent and we cannot apply greedy inference.

An important subclass of inference methods suitable for such complex inference problems are *incremental inference methods* (Collins and Roark, 2004; Daumé III and Marcu, 2005b; Daumé III et al., 2009; Ross et al., 2011; Doppa et al., 2014; Daumé III et al., 2014; Chang et al., 2015). Since we will use such methods frequently to obtain approximately optimal substructures in Chapter 6, we explain them in more detail in the following.

5.4.2.1 Issues with Non-incremental Inference

So far, we have assumed that inference consists in solving Equation 5.7,

$$f_\theta(x) = \underset{(h,z)\in\mathcal{H}_x\times\mathcal{Z}_x}{\arg\max} \langle\theta, \phi(x,h,z)\rangle$$

in just one step: We have to provide an algorithm that gets as input the parameter vector θ and the document x, and then outputs the highest-scoring latent structure. However, this can be infeasible if there are strong dependencies in the structure. Consider again the mention-entity model. In order to obtain an exact solution to Equation 5.7, we would have to enumerate and score all assignments of mentions to entities in

a document. Since the number of such assignments is exponential in the number of mentions in a document, solving the equation is infeasible.

5.4.2.2 Intuition

An option to tackle complex inference problems is to consider inference as an *incremental* task. Instead of enumerating all latent structures, the structures are constructed incrementally in steps $t = 1, 2, \ldots$. Each intermediate structure only models coreference relations for a subset of mentions. At a step t, the inference algorithm can access the structure constructed in step $t - 1$, which allows to take already established coreference relations into account.

Mention-entity models (e.g. Yang et al., 2008; Webster and Curran, 2014) typically follow this approach. The latent structure obtained in step $t - 1$ describes the partial entities for the mentions m_1 to m_{t-1}. In the next step t, we want to determine to which partial entity (if any) m_t belongs. Each attachment to a partial entity results in a latent structure, which then can serve as input for the next step.

5.4.2.3 Formalization

In order to describe incremental inference, we first need some new terminology. Given a set of latent structures $\mathcal{H}$ for some approach, let $\texttt{subgraphs}(\mathcal{H})$ denote the set of all subgraphs of the graphs in $\mathcal{H}$. Since the graphs in $\mathcal{H}$ encode coreference information for whole documents, the graphs in $\texttt{subgraphs}(\mathcal{H})$ encode partial coreference information.

Approaches based on incremental inference have as parameter a function

$$\texttt{Generate}\colon \texttt{subgraphs}(\mathcal{H}) \rightarrow 2^{\texttt{subgraphs}(\mathcal{H}) \times \mathcal{Z}} \tag{5.14}$$

that, given some directed labeled hypergraph corresponding to a partial latent structure at a time step t, outputs the set of candidate latent structures (together with the coreference relations encoded by them) for the next time step.

Incremental inference then operates as follows: starting from a dummy initial latent structure, iteratively apply the `Generate` function and choose the highest-scoring latent structure in the search space at each time step. Algorithm 5.1 formalizes this approach and Figure 5.4 shows a snapshot of the incremental inference procedure for a mention-entity model.

Algorithm 5.1. General incremental inference.

Input: A document $x \in \mathcal{X}$, a function `Generate`

1: **function** INCREMENTALINFERENCE(x, `Generate`)
2: Set h^0 to a dummy latent structure
3: Set $t = 0$
4: **while** `Generate`$(h^t) \neq (\emptyset, \emptyset)$ **do**
5: Set $(h^{t+1}, z^{t+1}) = \arg\max_{(h,z) \in \texttt{Generate}(h^t)} \langle \theta, \phi(x, h, z) \rangle$
6: Set $t = t + 1$

Output: The final latent structure h^t

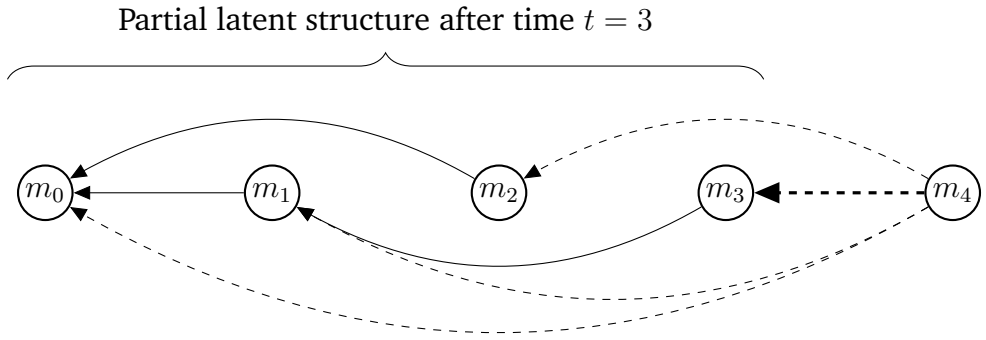

Figure 5.4: Incremental inference for a mention-entity approach. Each dashed edge induces a different latent structure. All these latent structures form the search space for the time step $t = 4$.

5.4.2.4 Incremental Inference as an Approximation Algorithm

Note that the output of Algorithm 5.1,

$$f_\theta(x) = \text{INCREMENTALINFERENCE}(x, \texttt{Generate}), \tag{5.15}$$

does not necessarily constitute an exact solution to Equation 5.7,

$$f_\theta(x) = \arg\max_{(h,z) \in \mathcal{H}_x \times \mathcal{Z}_x} \langle \theta, \phi(x, h, z) \rangle,$$

since the incremental inference procedure may apply state transitions that are locally optimal, but globally suboptimal.

To understand why, consider again the mention-entity model. Assume we are processing a mention m_5, the partial entities are $\{m_1, m_2\}$ and $\{m_3, m_4\}$. Attaching m_5 to $\{m_1, m_2\}$ results in a higher-scoring latent structure, therefore m_5 is attached to

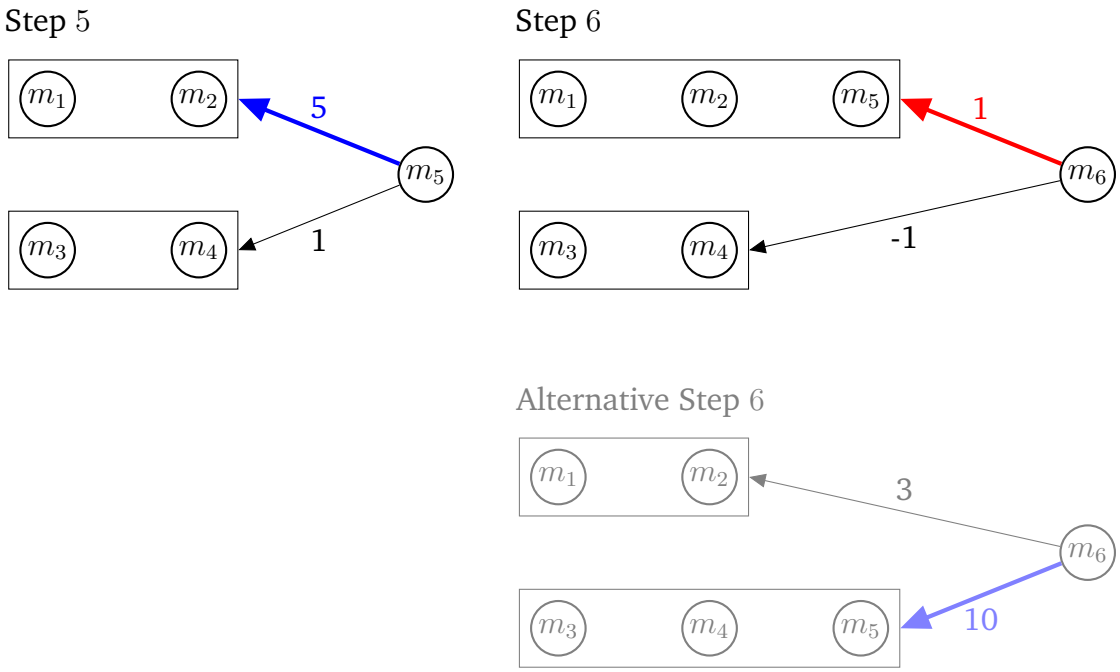

Figure 5.5: Globally suboptimal decisions during incremental inference. In step 5, attaching m_5 to $\{m_1, m_2\}$ yields the highest-scoring structure. Because of this decision, the partial entity $\{m_3, m_4, m_5\}$ is not in the search space for step 6. However, attaching m_6 to this partial entity would have led to the highest score.

$\{m_1, m_2\}$. Next, m_6 is processed and has access to the partial entities $\{m_1, m_2, m_5\}$ and $\{m_3, m_4\}$. Now it can happen that attaching m_6 to the partial entity $\{m_3, m_4, m_5\}$ would result in the highest score. However, m_6 does not have access to this partial entity, since the attachment of m_5 to $\{m_3, m_4\}$ was deemed suboptimal in the previous step. This example is visualized in Figure 5.5.

Hence, in order to guarantee optimality, we would need to consider all possible partial latent structures that can be constructed at each time step. Since the number of partial latent structures may be exponential in the number of mentions, this approach is not practical. The problem of globally suboptimal decisions is most severe when using greedy search as described in Algorithm 5.1. There exist approaches for refining search that try to avoid globally suboptimal decisions while ensuring reasonable time complexity and memory requirements, such as *beam search*. To do so, beam search keeps only the most promising solutions at each time step. However, compared to greedy search, beam search significantly increases running time and complicates pa-

rameter estimation (Bisani, 1987; Huang et al., 2012). We instead employ greedy search, as described in Algorithm 5.1, and use parameter estimation methods with strong performance guarantees (Chang et al., 2015).

5.4.2.5 Non-Incremental Inference as a Special Case

Non-incremental inference is a degenerated case of incremental inference. To see why, set $\texttt{Generate}(h^0) = (\mathcal{H}_x, \mathcal{Z}_x)$ and $\texttt{Generate}(h) = (\emptyset, \emptyset)$ for $h \neq h^0$. Then the while-loop in Algorithm 5.1 only gets executed once, and Algorithm 5.1 outputs $h = \arg\max_{(h,z)\in\texttt{Generate}(h^0)} \langle\theta, \phi(x, h, z)\rangle$. Since $\texttt{Generate}(h^0) = (\mathcal{H}_x, \mathcal{Z}_x)$, the output is the same as for non-incremental inference.

5.5 Parameter Estimation

Parameter estimation or *learning* means using some data to determine a parameter θ such that the predictor f_θ yields good performance on that data. We can broadly distinguish between *supervised learning*, where the data is annotated with the information we aim to learn (in our case coreference relations), and *unsupervised learning*, in which the data is not annotated with this information. In this thesis we only consider supervised learning, as it is the more popular paradigm when estimating parameters for coreference resolution approaches and also yields superior performance.

To learn the parameter vector, we have a *training set*

$$\mathcal{D} = \left\{(x^{(i)}, z^{(i)}) \mid i = 1, \ldots, k\right\} \subseteq \mathcal{X} \times \mathcal{Z} \tag{5.16}$$

at our disposal. $\mathcal{D}$ contains pairs $(x^{(i)}, z^{(i)})$ which consist of a document $x^{(i)}$ and the corresponding coreference annotation $z^{(i)}$. Note that the latent structures are not part of this training set: they are auxiliary structures, not annotated in the data and can differ per approach.

We now give perceptron-like algorithms that estimate a parameter vector $\theta \in \mathbb{R}^d$. We build upon a perceptron learning algorithm (Rosenblatt, 1958; Collins, 2002) – it is simple and fast, enables plug-and-play for different structures (providing respective decoders), can include task-specific cost functions via cost-augmented inference and has shown good performance for coreference resolution (Bengtson and Roth, 2008; Chang et al., 2012; Stoyanov and Eisner, 2012; Webster and Curran, 2014; Fernandes

et al., 2014). Furthermore, the perceptron can be extended to learn parameters in an incremental inference setting (Collins and Roark, 2004; Daumé III and Marcu, 2005b; Daumé III et al., 2014).

We first discuss a perceptron learning algorithm for the non-incremental case. We then give a learning algorithm for the incremental case and show that the non-incremental learning algorithm is a special case of the incremental learning algorithm.

5.5.1 Perceptron Learning: Non-incremental Case

Perceptron-like algorithms work by making a model prediction, and comparing this prediction with the expected output, as annotated in the training data. If the prediction was erroneous, components of the parameter vector are updated: components corresponding to features of the expected output are increased, while components corresponding to features of the prediction are decreased.

For the setting we discuss in this thesis, we need to employ some adaptations to the standard perceptron algorithm for structured prediction (Collins, 2002).

5.5.1.1 Coping with Latent Variables

In a setting with latent variables, we want to compare the latent prediction with some *reference* expected latent output. As such output is not available, since it is not part of the training data, latent perceptron algorithms compare the latent prediction with the highest-scoring latent prediction that is consistent with the reference annotation (Sun et al., 2009). We denote these predictions as $\hat{h}_{\text{cons}}$.

To define $\hat{h}_{\text{cons}}$ formally, we first introduce the latent output space restricted to structures encoding the reference annotations, which is

$$\mathcal{H}_{x,z} = \{h \in \mathcal{H}_x \mid h \text{ and } z \text{ are consistent}\} \subseteq \mathcal{H}_x \tag{5.17}$$

Then,

$$\hat{h}_{\text{cons}} = \arg\max_{h \in \mathcal{H}_{x,z}} \langle \theta, \phi(x, h, z) \rangle. \tag{5.18}$$

Departing from previous work, we allow to constrain the space $\mathcal{H}_{x,z}$. By doing so, we are able to account for the fact that some coreference approaches do not compare to the best prediction consistent with the reference annotation, but compare to a fixed structure which does not depend on the current parameter vector θ. For example, some

mention ranking models always compare to the closest correct antecedent (Denis and Baldridge, 2008).

Constraining the latent space is modeled by a function

$$\texttt{constrain}\colon 2^{\mathcal{H}} \rightarrow 2^{\mathcal{H}}. \tag{5.19}$$

This function maps $\mathcal{H}_{x,z}$ to $\texttt{constrain}(\mathcal{H}_{x,z}) \subseteq \mathcal{H}_{x,z}$. We mostly consider functions such that $|\texttt{constrain}(\mathcal{H}_{x,z})| = 1$: The latent output space is constrained to just one structure, against which we compare for training.

5.5.1.2 Cost-augmented Inference

As in previous work on the perceptron for coreference resolution (Chang et al., 2012; Fernandes et al., 2014), we employ *cost-augmented inference* (Crammer et al., 2006). Hence, when computing the model prediction for a training instance (x, z), we do not select the best-scoring output according to Equation 5.7. Instead, we solve the *cost-augmented problem*

$$f_\theta(x) = \underset{(h',z') \in \mathcal{H}_x \times \mathcal{Z}_x}{\arg\max} \langle \theta, \phi(x, h', z') \rangle + c(x, h', z), \tag{5.20}$$

where

$$c\colon \mathcal{X} \times \mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{R}_{\geq 0} \tag{5.21}$$

is a *cost function* that measures the cost of predicting h'. c has the property that $c(x, h', z) = 0$ if and only if h' is consistent with z . Predictions computed via Equation 5.20 maximize the sum of (i) $\langle \theta, \phi(x, h', z') \rangle$ and (ii) $c(x, h', z)$. Hence, ideally, they score well under the current model (according to (i)) and have high cost (according to (ii)). Since predictions that have high cost are highly incorrect, solving the cost-augmented problem pushes the learning algorithm towards predicting high-scoring incorrect solutions, which leads to more aggressive updates of the parameter vector.

5.5.1.3 Formal Description of the Algorithm

Algorithm 5.2 shows a more formal description of our perceptron learning algorithm. It employs parameter vector averaging to avoid overfitting (Freund and Shapire, 1999;

Algorithm 5.2. Structured latent perceptron with cost-augmented inference.

Input: A training set $\mathcal{D}$, functions `sub` and `constrain`, a cost function c, the number of epochs n

function PERCEPTRON($\mathcal{D}$, `sub`, `constrain`, c, n)
  Set $\texttt{counter} = 0$
  Set $\theta = (0, \ldots, 0) \in \mathbb{R}^d$
  for epoch $= 1, \ldots, n$ **do**
    for $(x, z) \in \mathcal{D}$ **do**
      for each substructure space $\mathcal{H}_{x,i} \in \texttt{sub}(x)$ **do**
        $(\hat{h}, \hat{z}) = \underset{(h', z') \in \mathcal{H}_{x,i} \times \mathcal{Z}_x}{\arg\max} \langle \theta, \phi(x, h', z') \rangle + c(x, h', z)$
        $\hat{h}_{\text{cons}} = \underset{h' \in \texttt{constrain}(\mathcal{H}_{x,z,i})}{\arg\max} \langle \theta, \phi(x, h', z) \rangle$
        if $\hat{h}$ is not consistent with z **then**
          Set $\theta = \theta + \phi(x, \hat{h}_{\text{cons}}, z) - \phi(x, \hat{h}, \hat{z})$
        Set $\theta_{\text{sum}} = \theta_{\text{sum}} + \theta$
        Set $\texttt{counter} = \texttt{counter} + 1$
  Set $\theta = \theta_{\text{sum}} / \texttt{counter}$

Output: A parameter vector θ

Collins, 2002)[5].

5.5.2 Perceptron Learning: Incremental Case

In order to extend the perceptron learning algorithm to the incremental case, we employ *imitation learning* by making use of the *learning to search* paradigm (Daumé III et al., 2014). In our notation, the goal of learning in this paradigm is to estimate a parameter vector θ such that the *transition function*

$$t \colon \text{subgraphs}(\mathcal{H}) \to \text{subgraphs}(\mathcal{H}) \tag{5.22}$$

induced by

$$t(h) = \underset{(h', z') \in \text{Generate}(h)}{\arg\max} \langle \theta, \phi(x, h', z') \rangle \tag{5.23}$$

performs well. Hence, we learn how to perform the search that underlies the incremental inference procedure. We choose the learning to search paradigm for many of the same reasons we employ the perceptron learning algorithm: it is simple to imple-

[5] We also employ shuffling by choosing a random unprocessed substructure space at each step. However, this is not displayed for readability.

ment, comparatively fast, works with arbitrary complex structures and it is devised to handle task-specific cost functions (Daumé III et al., 2014). Furthermore, learning to search has been successfully applied to coreference resolution (Daumé III, 2006; Ma et al., 2014; Clark and Manning, 2015, 2016). Learning to search requires an underlying cost-sensitive classification algorithm. For this algorithm, we employ the perceptron method presented in the previous section. As we will show, the learning to search algorithm with perceptron learning generalizes the perceptron learning algorithm presented in the previous section.

5.5.2.1 Learning to Search

In order to learn a parameter vector θ such that the induced transition function works well, learning to search algorithms first apply a *roll-in* transition function: given an input $(x, z) \in \mathcal{X} \times \mathcal{Z}$, they run a transition function $\texttt{in}$ to completion, starting from a dummy latent structure h^0. This yields a sequence $S = \{h^0, \ldots, h^n\}$ of latent structures, where $h^{i+1} = \texttt{in}(h^i)$. h^n is a latent structure for the whole document. There are many options for $\texttt{in}$. For example, we could use the transitioning function induced by the learned parameter vector (Equation 5.23), or a *reference* transitioning function that incrementally builds structures that are consistent with the reference annotation.

For every $h^i \in S$, $i > 0$, learning to search algorithms then consider alternatives

$$(h', z') \in \texttt{Alternatives}(h^i) \subseteq \texttt{Generate}(h^{i-1}). \tag{5.24}$$

for the time step i. We consider structures in $\texttt{Alternatives}(h^i)$, which is a subset of $\texttt{Generate}(h^{i-1})$, because we want to impose constraints on the alternatives considered for h^i. For example, if h^i was obtained by adding an edge (m, n) to the partial structure h^{i-1}, we may only want to consider alternatives h' that differ from h^i by choosing a different antecedent for m, and not by choosing an antecedent for a different mention.

By comparing the alternatives in $\texttt{Alternatives}(h^i)$, we can learn what constitutes a good decision during incremental inference. To do so, each alternative $(h', z') \in \texttt{Alternatives}(h^i)$ has an associated cost $c(h') \in \mathbb{R}_{\geq 0}$. To associate a cost, a *roll-out* transition function $\texttt{out}$ is run to completion for each partial latent structure in $\texttt{Alternatives}$. This yields latent structures $h'_{\texttt{out}} = \texttt{out}(\ldots(\texttt{out}(h'))\ldots)$. For a fixed $(h', z') \in \texttt{Alternatives}(h^i)$, we then set the cost of h' to the difference of the cost of

$h'_{\texttt{out}}$ and the lowest-cost alternative, that is

$$c(h') = c(x, h'_{\texttt{out}}, z) - \min_{(h'',z'') \in \texttt{Alternatives}(h^i)} c(x, h''_{\texttt{out}}, z). \tag{5.25}$$

As for `in`, there are many options for the transitioning function `out`.

For every $i \in \{1, \ldots, n\}$, the obtained cost-sensitive examples

$$\big\{(h', c(h')) \mid (h', z') \in \texttt{Alternatives}(h^i)\big\} \tag{5.26}$$

now serve as input for the cost-augmented perceptron learning algorithm discussed in the previous section. In particular, we compute the highest-scoring cost-augmented partial structure,

$$\Big(\hat{h}, \hat{z}\Big) = \operatorname*{arg\,max}_{(h',z') \in \texttt{Alternatives}(h^i)} \langle \theta, \phi(x, h', z') \rangle + c(h'), \tag{5.27}$$

and the highest-scoring partial structure with minimal cost,

$$\Big(\hat{h}_{\text{min}}, \hat{z}_{\text{min}}\Big) = \operatorname*{arg\,max}_{\substack{(h',z') \in \texttt{Alternatives}(h^i), \\ c(h') = \min_{(h'',z'') \in \texttt{Alternatives}(h^i)} c(h'')}} \langle \theta, \phi(x, h', z') \rangle, \tag{5.28}$$

and then perform the perceptron update $\theta = \theta + \phi(x, \hat{h}_{\text{min}}, \hat{z}_{\text{min}}) - \phi(x, \hat{h}, \hat{z})$. We need to resort to structures with minimal cost instead of structures that are consistent with the reference annotation, since, depending on the choice of the roll-in transition function `in`, no partial latent structure considered during a time step may be consistent with the reference annotation.

Figure 5.6 visualizes one step during learning to search for a tree-based mention-entity model.

5.5.2.2 Avoiding Roll-outs

We will now analyze conditions under which we can obtain the highest-scoring latent structures $\hat{h}$ and $\hat{h}_{\text{min}}$ without running the roll-out transitioning function `out` to completion. If we can avoid roll-outs, learning will be much faster.

If the cost function c factors over the edges of the latent structure, we have for

$(h', z') \in \texttt{Alternatives}(h^i)$

$$c(h') = c(x, \hat{h}, z) = \sum_{e \in \hat{h}} c(x, e, z) = \sum_{e \in h'} c(x, e, z) + \sum_{e \in \hat{h}, e \notin h'} c(x, e, z) \tag{5.29}$$

where $\hat{h} = \texttt{out}(\ldots(\texttt{out}(h'))\ldots)$ and the sums are over edges in the latent structures. We define

$$s(h', \texttt{out}) = \sum_{e \in \hat{h}, e \notin h', \hat{h} = \texttt{out}(\ldots(\texttt{out}(h'))\ldots)} c(x, e, z), \tag{5.30}$$

which is the sum of costs of edges that are not in h'.

We can avoid roll-outs if

$$s(h', \texttt{out}) = s(h'', \texttt{out}) \text{ for all } h', h'' \in \texttt{Alternatives}(h^i), \tag{5.31}$$

because then the cost only depends on the latent structure which serves as input to the roll-out. We will discuss edge-factorizing cost functions that fulfill the condition expressed in Equation 5.31 in our discussion of latent structures for coreference resolution in Chapter 6.

5.5.2.3 Formal Description of the Algorithm

Algorithm 5.3 shows a formal description of our learning to search algorithm with perceptron learning. The presented algorithm is a generic learning to search algorithm as presented in Daumé III et al. (2014). Instantiations of the algorithm are obtained by choosing transition functions `in` and `out`. Depending on the choice of transition functions, the instantiations correspond to novel learning to search algorithms or to approaches from the literature such as Searn (Daumé III et al., 2009), DAgger (Ross et al., 2011) or LOLS (Chang et al., 2015). Variants of learning to search algorithms have been applied to learn parameters for incremental entity-centric coreference resolution systems (Daumé III, 2006; Ma et al., 2014; Clark and Manning, 2015, 2016). We go beyond previous work by integrating a generic learning to search algorithm in a machine learning framework for coreference resolution, and by applying the algorithm to a great variety of approaches based on various structures.

Algorithm 5.3. Learning to search with perceptron learning.

Input: A training set $\mathcal{D}$, functions `sub` and `constrain`, a cost function c, the number of epochs n, the functions `Generate` and `Alternatives`, transition functions `in` and `out`

function LEARNINGTOSEARCH($\mathcal{D}$, `sub`, `constrain`, c, n, `Generate`, `Alternatives`, `in`, `out`)
  Set `counter` $= 0$
  Set $\theta = (0, \ldots, 0) \in \mathbb{R}^d$
  for epoch $= 1, \ldots, n$ **do**
    for $(x, z) \in \mathcal{D}$ **do**
      for each substructure space $\mathcal{H}_{x,i} \in$ `sub`(x) **do**
        Set `update` $= (0, \ldots, 0) \in \mathbb{R}^d$
        Set h^0 to a dummy latent structure
        Set $S = (h_0)$
        Set $t = 0$
        while `Generate`$(h^t) \neq (\emptyset, \emptyset)$ **do**
          Append $h^{t+1} =$ `in`(h^t) to S
          Set $t = t + 1$
        for $t = 1, \ldots, |S|$ **do**
          Using `out`, compute $\{(h', c(h')) \mid (h', z') \in \text{Alternatives}(h^t)\}$
          Set $\left(\hat{h}, \hat{z}\right) = \underset{(h',z') \in \text{Alternatives}(h^t)}{\arg\max} \langle \theta, \phi(x, h', z') \rangle + c(h')$
          Set $\left(\hat{h}_{\text{min}}, \hat{z}_{\text{min}}\right) = \underset{\substack{(h',z') \in \text{Alternatives}(h^t), \\ c(h') = \min_{(h'',z'') \in \text{Alternatives}(h^t)} c(h'')}}{\arg\max} \langle \theta, \phi(x, h', z') \rangle$
          if $\hat{h}$ is not consistent with z **then**
            Set `update` $=$ `update` $+ \phi(x, \hat{h}_{\text{min}}, \hat{z}_{\text{min}}) - \phi(x, \hat{h}, \hat{z})$
        Set $\theta = \theta +$ `update`
        Set $\theta_{\text{sum}} = \theta_{\text{sum}} + \theta$
        Set `counter` $=$ `counter` $+ 1$
  Set $\theta = \theta_{\text{sum}}/$`counter`

Output: A parameter vector θ

5.5.2.4 Non-incremental Perceptron Learning as a Special Case

Note that Algorithm 5.3 reduces to Algorithm 5.2 for non-incremental inference. Let (x, z) be a training instance. For non-incremental inference, we set

$$\texttt{Generate}(h^0) = (\mathcal{H}_x, \mathcal{Z}_x) \tag{5.32}$$

and

$$\texttt{Generate}(h) = (\emptyset, \emptyset) \text{ for } h \neq h^0 \in \mathcal{H}_x. \tag{5.33}$$

Furthermore, we set $\texttt{Alternatives}(h) = (\mathcal{H}_x, \mathcal{Z}_x)$ for all $h \in \mathcal{H}_x$.

Therefore $S = (h^0, h^1)$ with $h^1 = \texttt{in}(h^0)$ for some transitioning function $\texttt{in}$. Hence, the for loop in Line 14 of Algorithm 5.3 only considers $t = 1$, and we have

$$\left(\hat{h}, \hat{z}\right) = \underset{(h',z') \in \texttt{Alternatives}(h^1)}{\arg\max} \langle \theta, \phi(x, h', z') \rangle + c(h') \tag{5.34}$$

$$= \underset{(h',z') \in \mathcal{H}_x \times \mathcal{Z}_x}{\arg\max} \langle \theta, \phi(x, h', z') \rangle + c(h') \tag{5.35}$$

and

$$\left(\hat{h}_{\text{min}}, \hat{z}_{\text{min}}\right) = \underset{\substack{(h',z') \in \texttt{Alternatives}(h^1), \\ c(h') = \min\limits_{(h'',z'') \in \texttt{Alternatives}(h^1)} c(h'')}}{\arg\max} \langle \theta, \phi(x, h', z') \rangle \tag{5.36}$$

$$= \underset{\substack{(h',z') \in \mathcal{H}_x \times \mathcal{Z}_x, \\ c(h') = \min\limits_{h'' \in \mathcal{H}_x} c(h'')}}{\arg\max} \langle \theta, \phi(x, h', z') \rangle \tag{5.37}$$

Now by the definition of c in Equation 5.25,

$$c(h') = c(x, h'_{\texttt{out}}, z) - \min_{(h'',z'') \in \texttt{Alternatives}(h^1)} c(x, h''_{\texttt{out}}, z) \tag{5.38}$$

$$= c(x, h', z) - \min_{(h'',z'') \in \mathcal{H}_x \times \mathcal{Z}_x} c(x, h'', z) \tag{5.39}$$

$$\tag{5.40}$$

By definition of cost functions, $c(x, h, z) \geq 0$ and $c(x, h, z) = 0$ if and only if h is consistent with z. Therefore, for any structure h' that is consistent with the reference annotation, $c(h')$ is minimal with $c(h') = 0$. For all other structures, $c(h') = c(x, h', z)$.

It follows that

$$\left(\hat{h}, \hat{z}\right) = \underset{(h',z')\in\mathcal{H}_x}{\arg\max}\langle\theta, \phi(x, h', z')\rangle + c(x, h', z) \tag{5.41}$$

and

$$\left(\hat{h}_{\text{min}}, \hat{z}_{\text{min}}\right) = \underset{\substack{(h',z')\in\mathcal{H}_x\times\mathcal{Z}_x,\\ c(h'')=0}}{\arg\max} \langle\theta, \phi(x, h', z')\rangle \tag{5.42}$$

$$= \underset{h'\in\mathcal{H}_{x,z}}{\arg\max}\langle\theta, \phi(x, h', z)\rangle. \tag{5.43}$$

Therefore, the update in Line 20 of Algorithm 5.3 corresponds to the standard perceptron update in Algorithm 5.2.

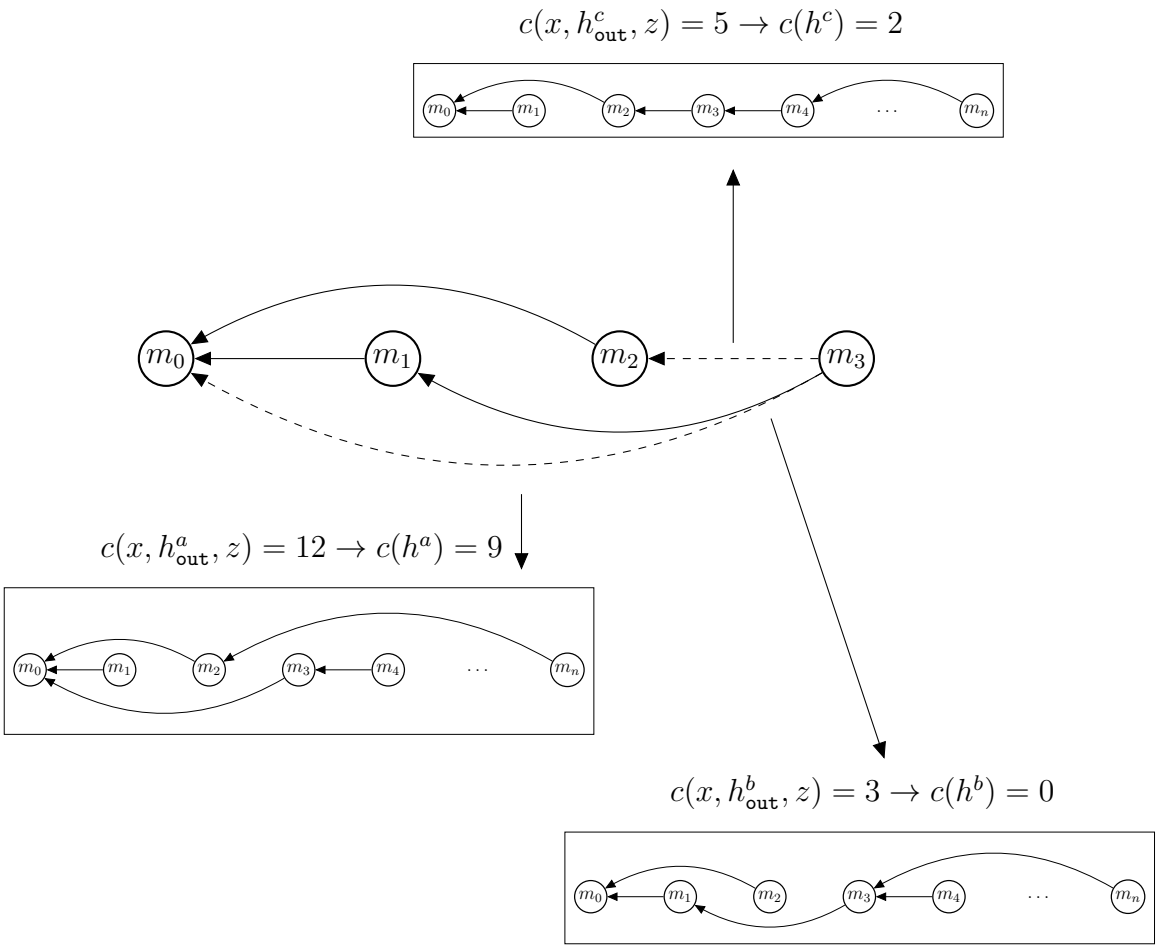

Figure 5.6: One step during learning to search for a mention-entity approach. The solid edges in the graph in the center were obtained by running the roll-in transitioning function until time step $t = 3$. Dashed edges induce alternatives $(h', z') \in \texttt{Alternatives}(h^3)$. For each alternative, we run the roll-out transitioning function to completion, and compute a cost for the complete structure. The cost associated to each alternative is then computed by subtracting the minimum cost over all alternatives. In the case displayed here, the weights would be updated by increasing weights of features of h_b and decreasing weights of features of h_a.

6 Structures for Coreference Resolution

In this chapter we discuss specific approaches to coreference resolution in detail and show how these can be expressed in the framework presented in Chapter 5. We discuss models proposed in the literature and devise novel model variants. We compare the models purely in terms of the structures they operate on. An extensive experimental evaluation and analysis is performed in Chapter 8.

We start the chapter with a general discussion of the approaches we consider (Section 6.1). We discuss three classes of approaches, depending on the structure they operate on: mention pair models, mention ranking models and antecedent trees, and entity-based models (Sections 6.2 to 6.4).

6.1 General Remarks

In Section 5.1 we observed that approaches to coreference resolution can be understood as predictors of *latent structures* between mentions. The machine learning framework presented in Chapter 5 draws upon and formalizes this observation. The framework enables us to give a uniform representation of approaches to coreference resolution. In particular, an approach to coreference resolution is defined by

- the space $\mathcal{H}$ of latent structures,
- the function `sub` for generating substructures,
- the function `constrain` for constraining the latent space of structures consistent with the reference annotation,
- the cost function c employed by the approach,

- the algorithm for obtaining high-scoring latent structures, also called the *decoder*, and
- the procedure `obtain_coreference` for obtaining coreference information z from predicted latent structures h.

For models that rely on complex latent structures with strong dependencies between the decisions encoded in the structure, we will employ incremental inference. For such models, we additionally need to describe the functions `Generate` and `Alternatives` that output candidate latent structures.

We now describe mention pair models (Soon et al., 2001; Ng and Cardie, 2002), mention ranking models and antecedent trees (Denis and Baldridge, 2008; Chang et al., 2012; Fernandes et al., 2014), and entity-based models (Luo et al., 2004; Yang et al., 2008; Stoyanov and Eisner, 2012) in terms of these parameters, highlighting differences and similarities between the approaches. We discuss variants of these models proposed in the literature, as well as novel variants.

6.2 Mention Pair Models

Mention pair models label each pair of mentions as *coreferent* or *non-coreferent*. Typically, such models do not enforce consistency of these individual decisions, which necessitates a clustering step to obtain coreference chains from the individual pair predictions.

6.2.1 A General Mention Pair Model

Most mention pair models share the same latent structure, but can differ in the remaining parameters, such as pruning of latent structures during training or the algorithm to obtain coreference information from latent structures. Different choices of these parameters correspond to different models from the literature, but can also lead to novel variants.

6.2.1.1 Parameters

We now go through each of the parameters discussed in Section 6.1, discussing their relationship to each other and discussing the models they lead to.

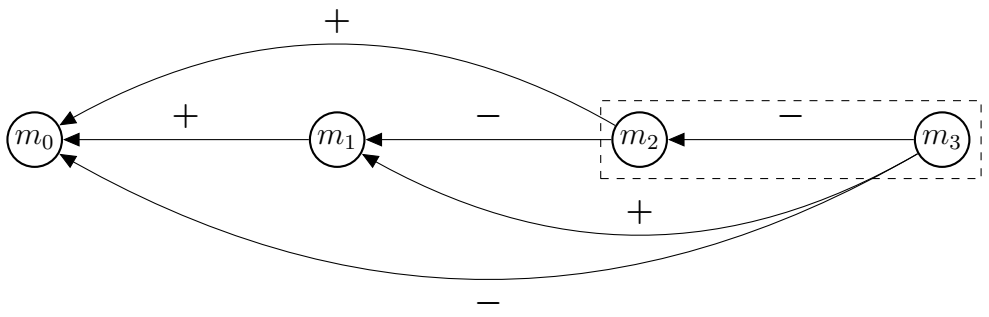

Figure 6.1: Graph-based representation of mention pair models. The dashed box shows one *substructure* of the structure.

Space $\mathcal{H}$ of latent structures. Most mention pair models label pairs of mentions individually as coreferent or non-coreferent. Hence, in terms of latent structures, they are based on *directed labeled graphs* that contain an edge between each pair of mentions. The label either denotes coreference or non-coreference. More formally, they are based on latent structures of the form $h = (M_x^0, A, L_A)$, where the set of edges contains each pair, that is

$$A = \left\{(m_j, m_i) \mid m_j, m_i \in M_x^0, j > i\right\}. \tag{6.1}$$

Coreference decisions are represented by edge labels for this graph. Each edge receives either the label "+" (coreferent) or "−" (not coreferent). Hence, L_A is a mapping

$$L_A \colon\ A \to \{+, -\}. \tag{6.2}$$

Note that the edges of the graph are fixed: two graphs in $\mathcal{H}_x$ only differ with respect to their edge labels. Figure 6.1 shows an example graph for a document x with mentions $M_x = \{m_1, m_2, m_3\}$.

Typically, mention pair models do not employ dummy mentions, but model detection of anaphoricity implicitly: if for some $j > 0$ none of the edges (m_j, m_{j-1}), …, (m_j, m_1) receives the label "+", m_j is deemed as non-anaphoric. Some approaches also employ an anaphoricity classifier in a preprocessing step. However, in this thesis we are interested in how we can model properties of the coreference resolution task structurally. We will not investigate anaphoricity classifiers further.

Most mention pair models employ heuristics to change the distribution of pairs in the training data, either to align training data creation with the clustering method

used or to cope with the fact that the non-coreferent pairs in A vastly outnumber the coreferent pairs (Soon et al., 2001; Ng and Cardie, 2002; Björkelund and Farkas, 2012). The most popular heuristic is proposed by Soon et al. (2001): let m_j be a mention. If m_j has no antecedent, disregard all pairs (m_j, m_i) with $i < j$. Otherwise, let m_k be the closest antecedent. Disregard all pairs (m_j, m_i) with $i < k$. In our framework, such a heuristic corresponds to pruning the graphs during training.

Substructure-generating function `sub`**.** Most mention pair models model the coreference decision for each pair as an individual prediction: each edge receives a label without depending on the label of any other edge. This suggests that each edge in the graph should correspond to one substructure. Formally, let A be the edge set of the mention pair model. We have

$$\texttt{sub}(x) = \{\mathcal{H}_{x,a} \mid a \in A\} \tag{6.3}$$

where $\mathcal{H}_{x,a}$ is $\mathcal{H}_x$ restricted to the edge a. That is, $\mathcal{H}_{x,a}$ only contains two graphs. Both graphs contain only the edge a. One graph labels this edge with "+", the other graph labels this edge with "−". In Figure 6.1, the dashed box shows one such substructure $s \in \mathcal{H}_{x,(m_3,m_2)} \in \texttt{sub}(x)$.

By modifying this substructure-generating function we can devise new variants of the mention pair model. For instance, by considering the graph for the whole document as a substructure, we consider all coreference relations in the document simultaneously. This makes a difference when learning parameters, since the parameter vector is not updated after each pair, but after each document.

Substructure-constraining function `constrain`**.** For mention pair models, there is exactly one latent structure that is consistent with the reference annotation. This is the latent structure h with the correct edge labeling: "+" for all edges that connect coreferent mentions, "−" for all edges that connect non-coreferent mentions. Therefore we cannot restrict the latent space of structures consistent with the reference annotation.

Cost function c. For mention pair models, the structured prediction task reduces to binary classification of mention pairs. For such classification tasks, using cost functions corresponds to resampling the training data under different distributions than the original distribution (Elkan, 2001; Geibel and Wysotzk, 2003).

As discussed above, most mention pair models employ heuristics to change the distribution of pairs in the training data. We are not aware of any work that uses cost functions for mention pair models.

Decoder. The decoder outputs the highest-scoring substructure under the current parameter vector. For computing the scores, mention pair models assume that the feature function factors according to the edges in the structure and that there are no dependencies between edge labels. Then, the highest-scoring substructure can be computed by greedy inference, where for each edge the label is chosen that leads to the highest score for the edge.

Obtaining coreference information from latent structures. The predicted latent structure is a graph such as the one displayed in Figure 6.1. Mention pairs connected by an edge with label "+" are deemed as coreferent by the model, while pairs connected by an edge with label "−" are deemed as not coreferent. Consequently, we need to coordinate these decisions to obtain the coreference chains, and there are many options for coordinating the decisions.

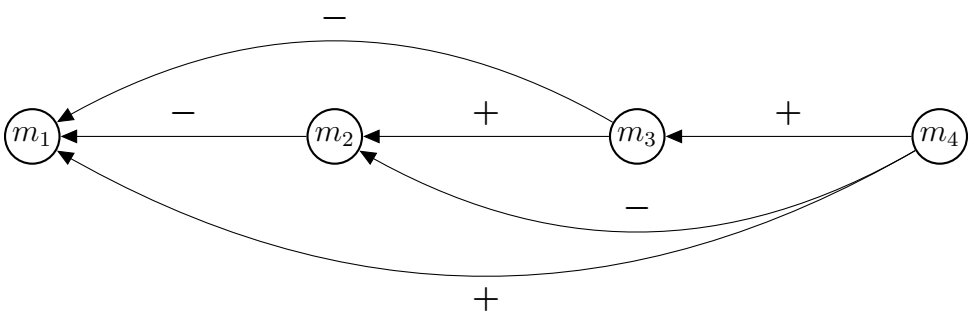

Figure 6.2: A more complex mention pair example.

Consider the example displayed in Figure 6.2 (we left out the dummy mention m_0 for readability). According to the prediction, m_4 is coreferent with m_1 as well as with m_3, but non-coreferent with m_2. However, the predictions deems m_3 and m_1 as non-coreferent and m_3 and m_2 as coreferent. How can we handle this contradicting information?

This corresponds to the clustering problem for mention pair models discussed in Section 3.1. A range of solutions has been proposed, from simple greedy schemes to

complex clustering approaches incorporating transitivity. Most of the methods rely on the scores of the edges according to the model, that is

$$\text{score}(m_j, m_i, \ell) = \langle \theta, \phi(x, (m_j, m_i, \ell), z) \rangle \in \mathbb{R}, \tag{6.4}$$

where $\ell \in \{+, -\}$. We therefore assume that we have access to these scores when we want to obtain coreference information from predicted latent structures. Popular methods include:

- *Closest-first* (Soon et al., 2001): for a mention m_j, let m_i be the closest preceding mention such that $L_A(m_j, m_i) = +$. Build the transitive closure over all such pairs (m_j, m_i).

- *Best-first* (Ng and Cardie, 2002): for a mention m_j, let m_i be the highest-scoring preceding mention labeled coreferent, i.e.

$$m_i = \underset{m_k \text{ s.t. } L_A(m_j, m_k)=+}{\arg\max} \text{score}(m_j, m_k, +). \tag{6.5}$$

  Build the transitive closure over all such pairs (m_j, m_i).

- *Aggressive Merge* (McCarthy and Lehnert, 1995): Perform the transitive closure for all pairs (m_j, m_i) that are labeled as coreferent.

Researchers found it beneficial to align the method with the resampling heuristic for creating training data (Ng and Cardie, 2002, p. 106f.). For instance, when employing *Aggressive Merge*, we should not apply any resampling. If we, however, employ *Closest First*, we should learn only from the closest correct antecedent of each mention. We will examine this experimentally.

There are many more choices for coordinating the decisions, most notably graph-based approaches (Nicolae and Nicolae, 2006; Cai and Strube, 2010a; Sapena et al., 2013) and approaches relying on combinatorial optimization (Klenner, 2007; Finkel and Manning, 2008), but we do not discuss these further, since the above mentioned methods are the most popular. All methods for coordinating the decision can be considered as implementations of the `obtain_coreference` function to extract coreference information from latent structures.

6.2.1.2 Discussion

Mention pair models constitute an attractive starting point for developing approaches to coreference resolution. They find a simple model for the task by reducing the problem to binary classification of mention pairs. In our framework, the implementation of such models is straightforward by relying on labeled graphs with edges between all pairs of mentions.

Mention pair models pay a price for reducing the problem to a simple setting: due to the complexity of the task, the reduction necessitates a post-processing clustering step.

Some mention pair models enforce that the decisions made by the model are consistent with each other, both during learning and prediction (Chang et al., 2011; Song et al., 2012). To model this in our framework, we need to modify the set of valid latent structures such that these are not able to represent contradicting information.

To do so, we require L_A to satisfy

$$L_A(m_j, m_i) = + \text{ and } L_A(m_i, m_k) = + \text{ implies } L_A(m_j, m_k) = + \tag{6.6}$$

for any $i, j, k \in \{1, \ldots, n\}$. Hence, the edge labeling is always transitive with respect to coreference, and therefore there can not be any contradictions. However, finding optimal structures in this setting requires solving a complex combinatorial optimization problem.

Mention ranking and antecedent tree models, which we consider in the next section, instead enforce consistency by modeling single-antecedent constraints *structurally* via considering an edge set A different from mention pair models. As we will see, this will lead to a simpler modeling approach which still permits greedy inference.

6.3 Mention Ranking and Antecedent Trees

Mention ranking and antecedent tree models cast coreference resolution as a *ranking* problem: given an anaphor m_j, they consider the list of candidate antecedents $m_0, \ldots, m_{j-1}$. The idea is to learn a parameter vector that enables to pick a correct antecedent m_i from this list. In other words, there exists a correct antecedent that is *ranked* higher than any incorrect antecedent. Since every mention gets assigned to only one antecedent, there is no need for a complex `obtain_coreference` method.

Coreference chains are obtained by transitive closure over all anaphor-antecedent decisions.

In this section, we discuss several approaches that built upon this idea. In order to be able to do efficient inference, the models assume that both the features and the cost-functions are *edge-factored*: there are no features or cost function computations that consider more than one edge. When extending the scope of the feature or the cost functions, the resulting models are *mention-entity* or *entity-entity* models, which we will discuss in the next section.

6.3.1 Mention Ranking

Mention ranking models stay closest to the idea just described: for each anaphor, pick the highest-ranked antecedent, where higher scores for anaphor-antecedent pairs correspond to higher ranks of the antecedent.

6.3.1.1 Parameters

Space $\mathcal{H}$ of latent structures. The latent structure underlying the mention ranking model represents the antecedent decisions for each anaphor in the document. This can be modeled as an unlabeled *tree*, where each mention node (except for the dummy mention m_0) has exactly one outgoing edge. Formally, ranking models are based on latent structures of the form $h = (M_x^0, A, L_A)$, where A is a subset of all mention pairs, $A \subseteq \{(m_j, m_i) \mid j > i\}$, that satisfies that each node has exactly one outgoing edge:

$$\text{for all } j > 0, \text{ there exists exactly one } i < j \text{ s.t. } (m_j, m_i) \in A. \tag{6.7}$$

The graph models the antecedent decisions via the structure, and does not need any edge labels, therefore $L_A(m_j, m_i) = \texttt{None}$ for all $(m_j, m_i) \in A$.

Figure 6.3 shows an example graph for a document with mentions $M_x = \{m_1, m_2, m_3\}$. According to this graph, m_1 and m_2 have as antecedents the dummy mention — they are not anaphoric — and m_3 has m_1 as antecedent.

Note that the edges have a different semantics than the edges in the representation of the mention pair model. While for mention pair models labeled edges signal coreference or non-coreference, edges now signal that the connected mentions are in an anaphor-antecedent relation. The absence of an edge does not entail that the mentions are not coreferent.

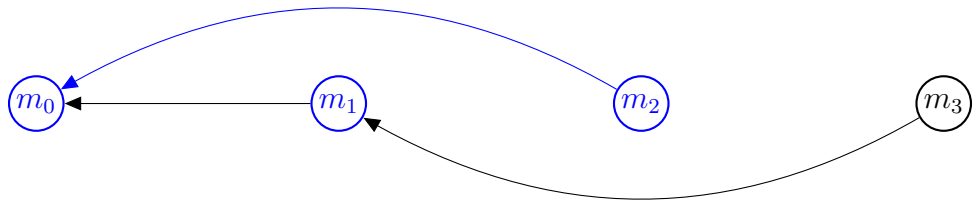

Figure 6.3: Graph-based representation of the mention ranking approach. A substructure is highlighted in blue.

Early approaches to mention ranking did not explicitly make use of dummy mentions (Denis and Baldridge, 2008). Since by construction mention ranking approaches find an antecedent for every anaphor, models without dummy mentions need to apply an anaphoricity classifier beforehand. For mentions that are deemed non-anaphoric by the classifier, the ranking model will not attempt to determine an antecedent. In this thesis, we will only consider ranking models with dummy mentions, since dummy mentions provide a way to structurally model anaphoricity detection.

Early approaches also applied similar heuristics as mention pair models to change the distribution of the training data. Equivalently to mention pair models, we can model these by pruning the graphs during training.

Substructure-generating function `sub`**.** The distinctive feature of the mention ranking approach (compared to antecedent trees) is that it models the antecedent decision for each anaphor individually. Hence, the jth generated substructure space by `sub` consists of all graphs in $\mathcal{H}_x$ that model the antecedent decision for m_j: $h = (V, A, L_A) \in \mathcal{H}_{x,j}$ if and only if

- $V = \{m_0, \ldots, m_j\}$,
- there exists exactly one m_i, $i < j$, such that $A = \{(m_j, m_i)\}$, and
- $L_A(m_j, m_i) =$ `None` for this (m_j, m_i).

In particular, each substructure consists of a graph that has only one edge. In Figure 6.3, such a substructure is highlighted in blue.

As we will see in the next subsection, extending the substructure to the whole document leads to antecedent trees (Fernandes et al., 2014). Hence, antecedent trees can be regarded as an extension of the mention ranking approach to the document level.

Substructure-constraining function `constrain`**.** In general, given a document $x \in \mathcal{X}$, there exist many latent structures for the mention ranking model that are consistent with the reference annotation. For example, consider a document with mentions $m_1, \ldots, m_5$. Suppose that m_2, m_3 and m_5 are coreferent. When determining an antecedent for m_5, there are two latent substructures consistent with the reference annotation. One substructure contains the link (m_5, m_3), the other contains the link (m_5, m_2).

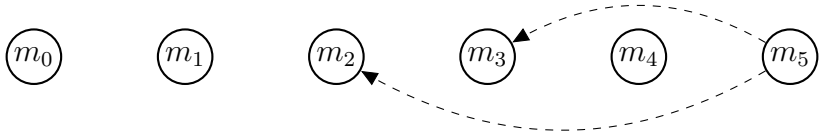

Figure 6.4: Latent substructures consistent with the reference annotation for the mention ranking model. If m_5 is coreferent with m_3 and m_2, both dashed edges are valid choices for a latent structure consistent with the reference annotation.

Figure 6.4 visualizes both substructures. When not constraining the substructures, the decoder will choose the highest-scoring substructure consistent with the reference annotation, which may be the substructure containing the edge (m_5, m_2). However, some implementations of the mention ranking model, for instance Denis and Baldridge (2008), do not consider the highest-scoring antecedent when updating their model, but always consider the *closest* antecedent. In the example this would correspond to taking the substructure which has (m_5, m_3) as the edge. To model this in our framework, we constrain the set of valid latent structures consistent with the reference annotation to structures containing only links to the closest antecedent

Formally, we define the function `constrain` as follows. `constrain`$(\mathcal{H})$ is the set of all latent structures in $\mathcal{H}$ such that the edges are only between a mention and its closest correct antecedent.

Cost function c**.** Cost functions for mention ranking models measure the quality of the prediction by reviewing properties of the predicted edge. The simplest cost functions check whether the predicted edge contains coreferent mentions or not, and assigns a cost $\lambda > 0$ if the mentions are not coreferent (Chang et al., 2012). More sophisticated cost functions provide a finer distinction of the error made. Durrett and Klein (2013), for example, distinguish between three types of error, where each type

has a different cost: a *wrong link* error predicts that two non-coreferent mentions are coreferent, a *false new* error predicts that an anaphoric mention is non-anaphoric, and a *false anaphoric* error predicts that a non-anaphoric mention is anaphoric. In Chapter 8 we will evaluate the contribution of a similar cost function experimentally.

Decoder. The mention ranking model relies on a simple structure: the predicted substructure contains only one edge, which signals an anaphor-antecedent relationship for the anaphor in focus. The decoder works in a greedy way: given a substructure space $\mathcal{H}_{x,j}$, score all edges (m_j, m_i) for $i = \{0, \ldots, j-1\}$. Choose the highest-scoring edge, breaking ties by favoring mentions closer to m_j. For cost-augmented inference during training, add the cost to the score of each pair. For predicting the highest-scoring latent substructure consistent with the reference annotation, restrict the edge set to pairs (m_j, m_i) such that m_j and m_i are coreferent, or, if m_j is non-anaphoric, m_i is the dummy mention m_0.

Obtaining coreference information from latent structures. The predicted latent structure for the whole document represents all anaphor-antecedent decisions for the document. Hence, to obtain coreference information, we perform the transitive closure over the anaphor-antecedent pairs represented in the structure. We do ignore all pairs (m_j, m_0), i.e. where the antecedent is the dummy mention, since having the dummy mention as antecedent corresponds to the prediction that the mention is non-anaphoric.

6.3.1.2 Discussion

The mention ranking approach models anaphor-antecedent decisions structurally, by predicting a graph that contains all anaphor-antecedent decisions for a document. The approach considers each anaphor individually.

By employing a *structural* representation for anaphor-antecedent decisions, the underlying structure, as well as the decoder and the `obtain_coreference` function are very simple. Since we take the highest-scoring edge (m_j, m_i) when predicting a latent substructure for the mention m_j, the mention ranking model can also be understood as a way to integrate *best-first clustering* during training.

6.3.2 Antecedent Trees

Antecedent trees, originally proposed by Yu and Joachims (2009), gained popularity after the antecedent-tree-based approach of Fernandes et al. (2014) won the CoNLL-2012 shared task on coreference resolution (Pradhan et al., 2012; Fernandes et al., 2012). In contrast to mention ranking models, which consider each anaphor in isolation, antecedent tree models predict the antecedents for all mentions in a document simultaneously. While antecedent tree models can be obtained by a simple modification of mention ranking models, we nevertheless discuss them in detail due to their popularity.

6.3.2.1 Parameters

Space $\mathcal{H}$ of latent structures. Antecedent trees are based on the same latent structure as the mention ranking approach: a graph that encodes all anaphor-antecedent decisions, such as the one displayed in Figure 6.3.

Substructure-generating function `sub`**.** Antecedent trees differ from the mention ranking approach only with respect to the factorization into substructures. While the mention ranking approach considers each anaphor in isolation, and therefore consider per-anaphor substructures, the antecedent tree approach considers the whole document at once. Therefore we have

$$\texttt{sub}(x) = \{\mathcal{H}\}\,. \tag{6.8}$$

Substructure-constraining function `constrain`**.** For the `constrain` function, the same discussion as for mention ranking models applies here.

Cost function c. Remember that we assume that the cost function factors over the edges in the tree (we study more advanced cost functions for entity-based models). Hence, to obtain cost functions for antecedent trees, we extend cost functions for the mention ranking model. Let c be a cost function for the mention ranking model. To extend c to antecedent trees, we factor the tree into substructures, as in the mention ranking model, and compute the cost according to c for each substructure. To obtain the cost for the whole tree, we compute the sum over the costs for the substructures.

All existing implementations use such cost functions (Yu and Joachims, 2009; Fernandes et al., 2014; Björkelund and Kuhn, 2014).

Decoder. Due to the edge factorization of the features and the costs the highest-scoring tree can be obtained by greedy inference: for each mention m_j, $j > 0$, add the highest-scoring outgoing edge (m_j, m_i) to the tree.

Obtaining coreference information from latent structures. Since the mention ranking model and antecedent trees share the same latent structure, we obtain coreference information from antecedent trees with the same method as for the mention ranking model.

6.3.2.2 Discussion

Antecedent trees are a simple extension of mention ranking models. In particular, they share the same latent structure, the only difference is the factorization into substructures. Therefore, if the cost function and the features factor according to the edges in the tree, antecedent trees have the same expressive power as mention ranking models.

6.3.3 Beyond Trees

Mention ranking models and antecedent trees rely on a *tree* as a latent structure, which encodes all anaphor-antecedent decision in a document. This can also be understood as accounting for the *best-first* clustering approach in the structure.

Therefore, other structures correspond to other clustering schemes. In this subsection, we study in detail the approach when we replace *trees* with general directed *graphs* as the latent structure.

6.3.3.1 Parameters

Space $\mathcal{H}$ of latent structures. The underlying latent structure now encodes all coreference relations from a mention to all preceding mentions. A mention either has the dummy mention as antecedent or has one or more non-dummy antecedents. Hence, latent structures have the form $h = (M_x^0, A, L_A)$ with

$$A \subseteq \{(m_j, m_i) \mid j > i\} \tag{6.9}$$

such that

$$(m_j, m_0) \in A \text{ implies } (m_j, m_i) \notin A \text{ for all } i < j \tag{6.10}$$

and $L_A((m_j, m_i)) = \texttt{None}$ for all $(m_j, m_i) \in A$.

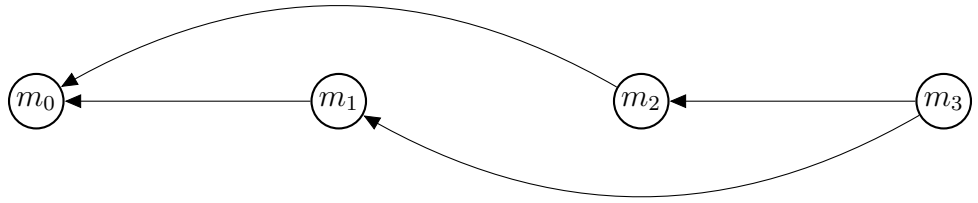

Figure 6.5: Graph-based representation of the approaches relying on general directed graphs instead of trees.

Figure 6.5 shows an example graph for a document with mentions $M_x = \{m_1, m_2, m_3\}$. In this example graph, m_3 has two antecedents: m_1 and m_2.

Hybrid approaches are also possible. For instance, we could allow all preceding coreferent mentions for non-pronominal mentions and only consider a single antecedent for pronouns.

Substructure-generating function `sub`**.** Similar to the distinction between the mention ranking approach and antecedents trees, natural definitions of `sub` are either to consider the induced substructures for each anaphor m_j, or to consider the whole document at once.

Substructure-constraining function `constrain`**.** When we do not constrain the structure consistent with the reference annotation, we learn from all correct coreference links to all preceding mentions.

With constraining the structure we can add some linguistic intuitions to the approach. For instance, we could impose distance restrictions or disregard links with specific mention types (such as proper name anaphor and pronoun antecedent).

Cost function c**.** Instead of just one edge for each anaphor, which represents an anaphor-antecedent relation, the graph-based approaches can predict more than one edge. Hence, we can obtain cost functions by extending cost functions devised for

mention ranking models to more than one edge, analogously to the extension to antecedent trees.

Decoder. Since we assume that the cost function and the features factor over the edges of the graph, we can obtain the highest-scoring graph by adding all edges with a positive score. If for an anaphor m_j the highest-scoring edge is (m_j, m_0), we only add this edge. If for an anaphor m_j no edge has a positive score, we add (m_j, m_0).

Obtaining coreference information from latent structures. In the graph-based models, two mentions are coreferent when there is an edge between the mentions. Therefore, to obtain coreference information, we take the transitive closure over all mention pairs which are connected by an edge (ignoring the dummy mention).

6.3.3.2 Discussion

Tree-based models such as mention ranking or antecedent trees rely on anaphor-antecedent relations and the single antecedent constraint: every mention has exactly one antecedent (which can be the dummy mention). Graph-based representations allow multiple antecedents for each mention, and therefore are based on the assumption that every mention has at least one antecedent. This assumption may be more reasonable than the single antecedent constraint for specific mentions which often lack strong anaphor-antecedent relations, such as for instance proper names (see the discussion in Section 2.1.2.2). Graph-based models can be understood as accounting for *aggressive-merge* clustering in the structure.

Furthermore, *hybrid* models are conceivable. For example, these may allow exactly one antecedent for pronouns, but multiple antecedents for proper names and common nouns.

6.4 Entity-based Models

Entity-based models (Luo et al., 2004; Rahman and Ng, 2011a; Björkelund and Kuhn, 2014; Clark and Manning, 2015, inter alia) leverage information about previous coreference predictions in a document. To do so, they incrementally construct coreference chains, which allows them to access the *partial entities* constructed so far. This enables

the use of *entity-level* features that examine properties of entities, such as the distribution of mention types or the structure of coreference chains (we describe the features used in our experiments in detail in Chapter 7).

Entity-based models can differ with respect to the latent structure they use, the exact algorithm employed for incrementally building structures and the factorization and scope of features and cost functions. Different instantiations of these parameters lead to different entity-based models. In this section, we describe two different latent structures on which approaches from the literature are built: trees/graphs and hypergraphs.

6.4.1 Tree and Graph Models

We first discuss entity-based models built on antecedent trees or graphs. By constructing these trees and graphs incrementally, decisions in later stages of inference can rely on previous anaphor-antecedent decisions. These entity-based models, which became very popular recently (Stoyanov and Eisner, 2012; Björkelund and Kuhn, 2014; Clark and Manning, 2015), retain anaphor-antecedent information between individual mentions while being able to employ entity-level information.

6.4.1.1 Parameters

Space $\mathcal{H}$ of latent structures. The latent structure these models are based on is the same as the structures discussed in Section 6.3, i.e. antecedent trees or antecedent graphs. While the structures are the same, the difference is that entity-based methods construct the structures *incrementally*, which allows them to access information about partial entities. In order to access detailed information about the incremental construction, we label each edge in the latent structure with the time step when it was added to the graph, that is

$$L_A((m_j, m_i)) = t \tag{6.11}$$

if and only if the edge (m_j, m_i) was added to the graph in time step t.

Note that entity-based models with trees/graphs as the underlying structure do not model partial entities explicitly: the latent structures are constructed by successively applying anaphor-antecedent decisions for pairs of mentions. Entity-based information can be injected into the models by devising entity-based features that examine properties of the tree/graph beyond the edge in focus. Another option to employ

entity-based information is via the cost function, as we will describe later.

Candidate-generating functions `Generate` and `Alternatives`. In entity-based models there are strong dependencies between individual coreference decisions. Therefore these models use incremental inference and learning to search. We have to specify the candidate-generating functions `Generate` and `Alternatives`. These functions receive as input a partial latent structure $h \in \texttt{subgraphs}(\mathcal{H})$. `Generate` outputs all latent structures to be considered in the next step of incremental inference, while `Alternatives` outputs alternative latent structures for the current time step during learning to search (see Section 5.5.2)[1].

The `Generate` function. Let us first consider the `Generate` function. We describe a general function. As we will show later, different constraints on this general function will lead to different inference schemes for entity-based models.

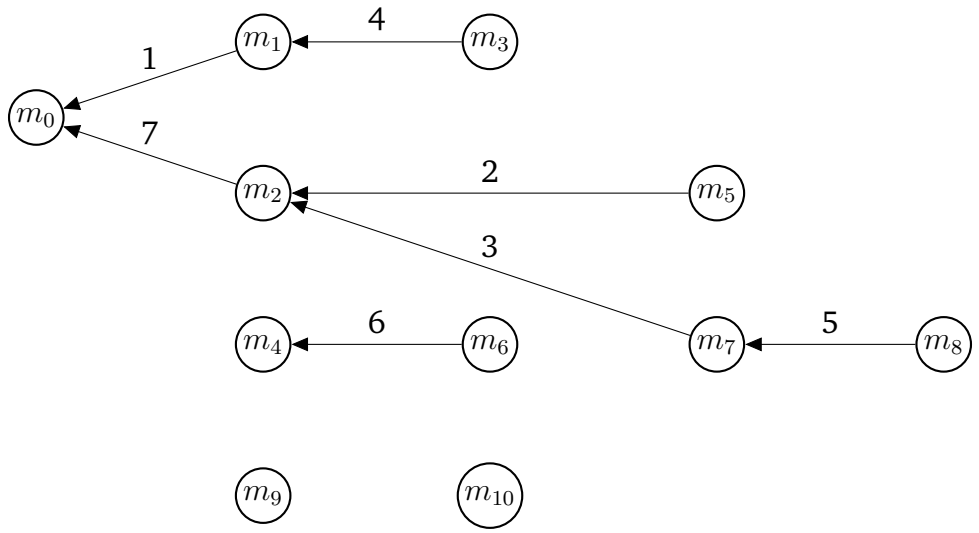

Figure 6.6: A partial latent structure for an entity-based model with trees as underlying structure. The partial latent structure is a forest, i.e. a graph where each connected component is a tree. Each tree corresponds to a partial entity. Edge labels denote the time step in which the edge was added to the tree. The mentions m_4, m_9 and m_{10} are unattached.

Partial latent structures are antecedent trees (respectively graphs) for subsets of

[1] In addition to the structures, these functions also output the coreference relations encoded by the structures. For convenience, we drop this output when describing the functions in this chapter.

mentions. Figure 6.6 shows one such partial latent structure for a tree-based model. Given a structure $h = (M_x^0, A, L_A)$, candidate latent structures for the next time step are obtained by different antecedent choices for unattached mentions (i.e. mentions with no outgoing edge). Formally, we define for tree-based models

$$\texttt{Generate}(h) = \left\{h' = (M_x^0, A_{j,i}, L_{A_{j,i}}) \mid m_j \text{ is unattached}, i \in \{0, \ldots, j-1\}\right\} \quad (6.12)$$

with

$$A_{j,i} = A \cup \{(m_j, m_i)\} \quad (6.13)$$

and

$$L_{A_{j,i}}(a) = \begin{cases} L_A(a) & a \in A, \\ \max_{a' \in A} L_A(a') + 1 & a = (m_j, m_i). \end{cases} \quad (6.14)$$

For graph-based models, we consider an arbitrary number of antecedents, therefore in this case

$$\texttt{Generate}(h) = \{h' = (M_x^0, A \cup E, L_{A \cup E}) \mid m_j \text{ is unattached}, \\ E = \{(m_j, m_0)\} \text{ or } E \subseteq 2^{\{(m_j, m_i) \mid 0 < i < j\}}\} \quad (6.15)$$

where $2^{\{(m_j, m_i) \mid 0 < i < j\}}$ is the set of all outgoing edges from m_j pointing back to non-dummy mentions, and $L_{A \cup E}$ is defined analogously to the tree case.

Note that different latent structures can express the same partial entity. If a partial entity before processing m_j contains the mentions m_{i_1} to m_{i_k}, then all attachments (m_j, m_i) with $i \in \{i_1, \ldots, i_k\}$ lead to the partial entity consisting of mentions $\{m_{i_1}, \ldots, m_{i_k}, m_j\}$. However, the internal structure is different.

The `Alternatives` function. `Alternatives` outputs alternative latent structures for the current step during learning to search. The latent structure in the current time step t was obtained by choosing the highest-scoring $h \in \texttt{Generate}(h^{t-1})$. Compared to h^{t-1}, h contains additional edges $(m_j, m_{i_1}), \ldots, (m_j, m_{i_k})$ with $k \geq 1$. Each of these edges is labeled with the current time step t. `Alternatives` considers alternative attachments of the mention m_j – this is in contrast to the `Generate` function, which additionally chooses the mention to be attached.

Hence, let m_j be the source of a highest-labeled edge in h. We set $\texttt{Alternatives}(h)$

to the subset

$$\texttt{Alternatives}(h) \subseteq \texttt{Generate}(h^{t-1}) \tag{6.16}$$

such that the graphs in $\texttt{Alternatives}(h)$ only add attachments from m_j. h^{t-1} can be obtained from h by removing all edges with the highest label.

Inference Schemes. By definition of the incremental inference procedure, we pick the highest-scoring latent structure $h \in \texttt{Generate}(h^{t-1})$ in time step t of the inference procedure for one document. For the `Generate` function as discussed above, the latent structure h is adding the highest-scoring (i.e. most confident) anaphor-antecedent decision (m_j, m_i) to the graph, where the search space is over all antecedent decisions for unattached mentions. This inference paradigm is also known as *easy-first inference* (Goldberg and Elhadad, 2010; Stoyanov and Eisner, 2012), since decisions which are considered easy/reliable/confident by the model are preferred. However, easy-first models face a very large search space, which leads to high computational complexity. To cope with the large search space, models from the literature restrict the search space, following one of two options.

The first option is to consider the *first* unattached mention (with respect to document order) instead of *any* unattached mention when generating candidate latent structures (Björkelund and Kuhn, 2014; Webster and Curran, 2014). Hence, such models attach mentions in document order. They do not perform easy-first inference, but *left-to-right inference*. In the second option, easy-first inference is retained, but the size of the search space is limited by employing heuristics or thresholds (Stoyanov and Eisner, 2012; Clark and Manning, 2015). For example, Stoyanov and Eisner (2012) only consider proper name antecedents for proper name mentions. Similar restrictions apply to other mention types.

In our framework, all of these restrictions can be represented as considering restricted versions of the `Generate` function as defined above. That is, we consider functions $\texttt{Generate}_{\text{restr}}$ with

$$\texttt{Generate}_{\text{restr}}(h) \subseteq \texttt{Generate}(h) \tag{6.17}$$

for all $h \in \mathcal{H}$. We evaluate different restrictions when performing the experiments for entity-based models in Section 8.4.

Substructure-generating function `sub`**.** The incremental inference procedure just described aims to find an approximately optimal solution for the whole document. We therefore have no factorization into substructures.

Substructure-constraining function `constrain`**.** We are not aware of any approaches that constrain structures consistent with the reference annotation.

Cost function c**.** Models based on trees or graphs can use cost functions from the non-incremental approaches based on these structures without any modification. Note that these cost functions are edge-factored.

Cost functions that are not edge-factored can be obtained from coreference resolution evaluation metrics such as as MUC (Vilain et al., 1995) and B^3 (Bagga and Baldwin, 1998). Such cost functions can be applied as follows: given a partial latent structure $h' \in \texttt{Generate}(h)$, roll out to obtain a latent structure for the whole document, obtain coreference information from this latent structure and compute a coreference resolution evaluation metric score s. The cost for the latent structure is then set to $\lambda(1 - s)$ for some $\lambda > 0$. Via using such a cost function, entity-based information can be incorporated into a model without using any entity-specific features or graph representations.

Decoder. The decoder computes the highest-scoring partial latent structure in the search space. The score of a partial latent structure can be obtained by summing the scores of all edges in the structure. When scoring the edges, we have to distinguish between two sets of features (see Chapter 7): *local* features, which only consider a pair of mentions, and *non-local* features, which consider a pair and *additionally* its context in the structure. The non-local features can be further divided into features that consider the context of both mentions in the pair (m_j, m_i), and into features that only consider the context of the antecedent m_j. If non-local features only examine the context of the antecedent, the model is commonly called an *entity-mention* or *mention-entity* model (Yang et al., 2008). Models that examine both contexts are sometimes dubbed *entity-centric* (Clark and Manning, 2015). We call such models *entity-entity* models.

Since the search space is very large, it is prohibitively expensive to score all edges of all structures in the search space. However, a structure $h \in \texttt{Generate}(h^t)$ differs from h^t by adding one or more edges to the graph represented by h^t. Entity-based

approaches assume that the score of h can be obtained by adding the score of the newly added edges to the score of h^t. For doing this efficiently, the models assume that the score of an edge is unaffected by later coreference decisions.

Hence, for all $h \in \texttt{Generate}(h^t)$, the decoder scores the edges with the highest label, and adds this score to the cached score of h^t to obtain a score for h. It then selects the highest-scoring partial latent structure in the search space.

Obtaining coreference information from latent structures. The predicted latent structure is a tree or graph representing anaphor-antecedent decisions. Therefore coreference information can be obtained by transitive closure over all edges in the tree/graph (ignoring the dummy mention).

6.4.1.2 Discussion

Entity-based models built on trees or graphs are attractive because they allow the inclusion of entity-level features, while still modeling structure in terms of anaphor-antecedent decisions. Varying the scope of the features leads either to mention-entity or entity-entity models.

6.4.2 Hypergraph Models

Most entity-based approaches to coreference resolution do not model anaphor-antecedent relations as do the ranking and tree models. Instead, they treat the incrementally built partial entities as atomic units (Luo et al., 2004; Culotta et al., 2007; Yang et al., 2008; Ma et al., 2014). Scores of mention-entity or entity-entity pairs only depend on features over all mentions in the partial entities, such as the existence of a head match, or the minimum distance between the partial entities. In particular, such models employ an *explicit* representation of partial entities. In our framework, these models can be expressed via *hypergraphs*.

6.4.2.1 Parameters

Space $\mathcal{H}$ of latent structures. In an explicit representation, partial entities are represented as sets of mentions, and coreference decisions are represented as hyperedges

that connect these sets. This is modeled by setting the set of valid edges to

$$A = \{(X, Y) \mid X \subseteq M_x, Y \subseteq M_x, X \cap Y = \emptyset, \exists m \in X, n \in Y \text{ s.t. } m > n\} \\ \cup \{(X, \{m_0\}) \mid X \subseteq M_x\} \tag{6.18}$$

Most of our features rely on anaphor-antecedent relations (see Chapter 7). Hence, these features only have a value when there exists a mention $n \in Y$ that precedes a mention $m \in X$. We therefore defined A such that any pair $(X, Y) \in A$ fulfills this relation.

Most hypergraph-based models from the literature are *mention-entity models*: they do not consider pairs of partial entities, but attach single mentions to preceding partial entities. In our representation, this is modeled by cardinality constraints on the hyperedges $(X, Y) \in A$: to only allow attachments of mentions, we require that $|X| = 1$.

Again, we label each edge in the latent structure with the time step when it was added to the graph.

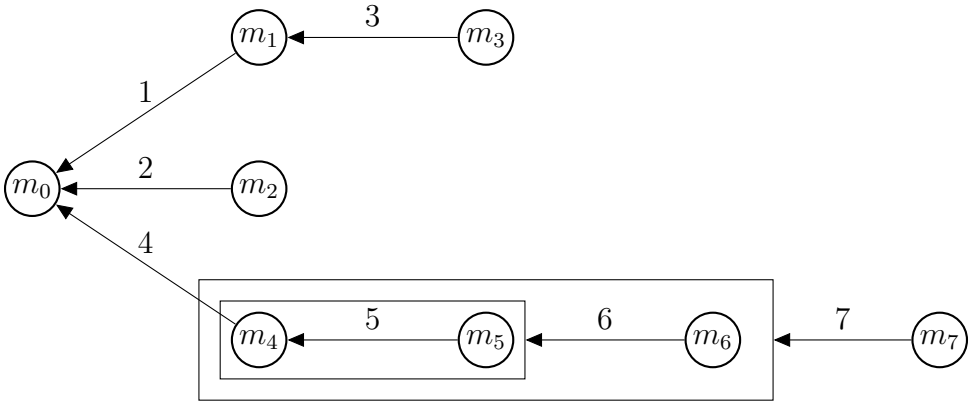

Figure 6.7: Hypergraph-based mention-entity model. The entities displayed are $\{m_1, m_3\}$, $\{m_2\}$ and $\{m_4, m_5, m_6, m_7\}$ The graph was constructed via mention-entity attachments m_1, m_2 and m_4 to $\{m_0\}$, m_3 to $\{m_1\}$, m_5 to $\{m_4\}$, m_6 to $\{m_4, m_5\}$ and m_7 to $\{m_4, m_5, m_6\}$.

Figure 6.7 shows an example hypergraph of a mention-entity model that encodes coreference decisions for a document with mentions $M_x = \{m_1, \ldots, m_7\}$.

Candidate-generating functions `Generate` and `Alternatives`. We only consider the `Generate` function. The `Alternatives` function can be obtained as described in

Section 6.4.1. The discussion of inference schemes from that section also applies to hypergraph models.

For hypergraph models, the `Generate` function obtains candidate latent structures by attaching partial entities to preceding partial entities. Formally, if $h = (M_x^0, A, L_A)$, we set

$$\begin{aligned}\texttt{Generate}(h) = \{h' = (M_x^0, A_{X,Y}, L_{A_{X,Y}}) \mid X \subseteq M_x, Y \subseteq M_x \text{ or } Y = \{m_0\},\\ X \cap Y = \emptyset, \exists (m,n) \in X \times Y \text{ s.t. } m > n, \nexists Z \text{ s.t. } (X,Z) \in A\}\end{aligned} \tag{6.19}$$

with

$$A_{X,Y} = A \cup \{(X,Y)\} \tag{6.20}$$

and

$$L_{A_{X,Y}}(a) = \begin{cases} L_A(a) & a \in A, \\ \max_{a' \in A} L_A(a') + 1 & a = (X,Y). \end{cases} \tag{6.21}$$

In the definition of $\texttt{Generate}(h)$, the condition $\nexists Z$ s.t. $(X,Z) \in A$ ensures that the partial entity represented by X is not already attached.

Substructure-generating function `sub`. The incremental inference procedure just described aims to find an approximately optimal solution for the whole document. We therefore have no factorization into substructures.

Substructure-constraining function `constrain`. We are not aware of any entity-based approaches relying on hypergraphs that constrain structures consistent with the reference annotation.

Cost function c. Analogously to tree/graph models, hypergraph models can use cost functions induced from coreference resolution evaluation metrics.

To obtain edge-factoring cost functions for hypergraph models, cost functions for ranking models can be adapted. We discuss two options. In the first option, we simply aggregate all costs between mention pairs induced by the pair of partial entities. Given a mention-ranking cost function c_{rank}, we set

$$c_{\text{aggr}}(x, (X,Y), z) = \sum_{m \in X, n \in Y, m > n} c_{\text{rank}}(x, (m,n), z). \tag{6.22}$$

This cost function is sensitive to the size of the partial entities. We also devise a variant that does not account for size of the partial entities. To do so, we take the maximum over all costs for induced mention pairs:

$$c_{\text{max}}(x, (X, Y), z) = \max_{m \in X, n \in Y, m > n} c_{\text{rank}}(x, (m, n), z). \tag{6.23}$$

To ensure that the cost functions only depend on the latent structure at the current time step, as required to avoid roll-outs, we set $c_{\text{aggr}} \equiv c_{\text{max}} \equiv 0$ for all edges except the one added in the current time step. Hence, it is sufficient to evaluate the cost for the edge added in the current time step. While learning with such cost functions is computationally efficient, the costs considered are only *local*: the effect of the decision on later decisions is not modeled.

Decoder. For all models, we assume that the features are defined over hyperedges. With this assumption, the decoder works analogously to the decoder of entity-based models relying on trees or graphs, which we described in the previous section.

Obtaining coreference information from latent structures. The output of the incremental inference procedure is a directed hypergraph such as the graph displayed in Figure 6.7. In this graph, two mentions (excluding the dummy mention) are deemed coreferent if they are in the same component after removing the dummy mention. Therefore, to obtain coreference information, we first remove the dummy mention and then assign two mentions to the same entity if they are in the same component.

6.4.2.2 Discussion

Entity-based hypergraph models regard coreference resolution as merging partial entities that exhibit no internal structure. This yields a model that is able to reason over sets of mentions, but is unable to express anaphor-antecedent relations or to compute features over individual mention pairs.

Many of the hypergraph-based approaches from the literature do not use learning to search during training, but rely on binary classification (Luo et al., 2004; Culotta et al., 2007; Yang et al., 2008). In our framework, this can be modeled by considering an edge-labeled variant of the hypergraph structures, analogously to the mention pair model. During training, the aim is to learn to predict correct labels for hyperedges. During test time, latent structures are built incrementally by greedily choosing

the highest-scoring hyperedge labeled with *coreferent*. Structurally, these models are very similar to mention pair models. Furthermore, in preliminary analysis we found that these models also suffer from the same weaknesses as mention pair models. We therefore do not consider these classification-based models further in this thesis.

7 Features

In this chapter we present the features employed by the approaches we will discuss in the next chapter. After giving an overview (Section 7.1), we discuss the various classes of features we use. We start with mention and mention pair features (Section 7.2), which are shared by all approaches. We then discuss features that are only applicable to entity-based models (Section 7.3). We conclude the chapter by presenting our feature combination scheme (Section 7.4).

7.1 Overview

If applicable, we employ the same set of mention and mention pair features in all models discussed in thesis. Our feature set consists of features commonly used in previous work (Bengtson and Roth, 2008; Durrett and Klein, 2013; Fernandes et al., 2014; Björkelund and Kuhn, 2014). Models such as mention-entity models have more representational power than models based solely on mention pairs. To make use of this gained representational power we devise features unique to these models.

Many of our features rely on the mention types of the constituting mentions, which we assign based on the part-of-speech tag of the mention's head. They are obtained by the following algorithm:

- if the head token has a named entity tag or if its part-of-speech tag starts with *NNP*, return *Proper Name*,
- if the part-of-speech tag starts with *PRP*, return *Pronoun*,
- if the part-of-speech tag starts with *DT*, return *Demonstrative Pronoun*,
- if the part-of-speech tag starts with *VB*, return *Verb*,
- otherwise, return *Miscellaneous*.

We follow previous work (Fernandes et al., 2014; Chang et al., 2012, 2013) and do not compute any features when the antecedent is the dummy mention m_0. Hence, anaphoricity determination is only modeled via the structure.

7.2 Local Features

Local features are features that only consider relations between two mentions in the latent structure. Hence, they operate on pairs (m, n) of mentions, where n precedes m. We call m the *anaphor* and n the *(candidate) antecedent*.

Type	Features
	Mention Features
Lexical & Surface	Head, first/last/preceding/following token, length
Knowledge	Head NE class, gender, number, semantic class
Grammatical	Fine-grained mention type
Syntactic	Dependency relation of head, governor, ancestry
	Pairwise Features
String Similarity	String match, head match, alias, tokens contained, head contained, modifier
Distance	Sentence distance, token distance
Miscellaneous	Embedded, same speaker

Table 7.1: Local features used in the models discussed in this thesis.

Table 7.1 shows an overview of the features. We distinguish between *mention features* and *pairwise features*.

7.2.1 Mention Features

Mention features examine the property of one mention. When extracting the features for a pair (m, n) of anaphor m and candidate antecedent n, all of the following features are extracted for both m and n.

7.2.1.1 Lexical and Surface Features

We first consider lexical features and features that can be obtained without any linguistic analysis. They are commonly used in current data-driven coreference resolution systems (Björkelund and Nugues, 2011; Durrett and Klein, 2013; Fernandes et al., 2014).

Head. Return the lowercased head of a mention. For a phrase, the head is the "central element which is distributionally equivalent to the phrase as a whole" (Chrystal, 2008, p. 225). For example, the head of *the man* is *man* and the head of *a quite interesting development* is *development*. With this feature, the model can learn associations from the training data. We describe how we extract heads in Section 8.1.

First/last/preceding/following token. Return the lowercased first, last, preceding and following tokens of a mention. Similar to the head feature, these tokens provide information to learn associations.

Length. Return the length of a mention in tokens. The length of a mention is correlated with information status, long mentions are less likely to have antecedents (Ariel, 1990).

7.2.1.2 Knowledge Features

Features from this category include named entity, gender, number and semantic class information. Extracting these features for anaphor and candidate antecedent allows to measure agreement of these values, which is especially helpful for pronoun resolution (Lappin and Leass, 1994).

Head NE class. If the mention is a proper name, return the named entity class (according to the data) of the last token of the mention's head. Otherwise, the value of this feature is *None*.

Gender. Return the gender of the mention, either *Male*, *Female*, *Neutral*, *Plural* or *Unknown*. For determining the gender of pronouns, we use a look up in a list. For proper names, we assign gender *Neutral* to non-person proper names. For person proper names, we first look for cue words such as *Mr.*. For common nouns and for

the remaining proper names we rely on a look up to the number and gender data of Bergsma and Lin (2006). We first look up the whole string of the mention. If this was not found in the data, we look up the whole head. If this was not found, we iteratively look up all upper case tokens of the head until one token is found in the data. If this was also not successful, we assign the value *Unknown*.

Number. Return the number of the mention, either *Singular*, *Plural* or *Unknown*. We assign number based on the part-of-speech tag of the mention's head.

Semantic Class. Return the semantic class of the mention, either *Person*, *Object*, *Numeric* or *Unknown*. For proper names, we map the named entity classes to the semantic classes. For common nouns we employ a WordNet (Fellbaum, 1998) lookup.

7.2.1.3 Grammatical Features

We only employ one grammatical feature, *fine-grained mention type*. This feature is important for our feature conjunction scheme, see Section 7.4.

Fine-Grained Mention Type. Return the fine-grained type of a mention. For proper names, we do not employ fine-grained types: each proper name has the feature value *Proper Name*. For common nouns, we distinguish between definite common nouns and other common nouns (*Definite* and *Non-Definite* respectively). For personal pronouns, we map each pronoun to its canonical form. For example, the feature value for a mention *him* is *he*. Demonstrative pronouns have the feature value *Demonstrative*. All other mentions receive the value *Miscellaneous*.

7.2.1.4 Syntactic Features

Syntactic features allow to capture the syntactic context of the mentions in their respective sentences. They also capture information about grammatical roles. Such information, as for example parallelism of grammatical roles, is considered helpful for pronoun resolution (Lappin and Leass, 1994).

Dependency Relation of Head. Return the dependency relation of the head to its governor. To obtain this relation, we convert all phrase structure parse trees shipped

with the data into Stanford dependencies (de Marneffe et al., 2006)[1].

Governor. Return the governor of the head.

Ancestry. Return the ancestry of the head as defined by Durrett and Klein (2013). Ancestry is defined as the part of speech tags of the tokens up to the grandparent in the dependency tree, concatenated with the direction taken in the tree (left or right).

7.2.2 Pairwise Features

Pairwise features examine relations between the mentions in a pair.

7.2.2.1 String Similarity Features

Features based on string similarity are of utmost importance for proper names and common nouns, since for such mentions string similarity is a strong indicator for coreference (e.g. Soon et al. (2001), see also Section 2.1).

String Match. Return whether the lowercased surface strings of anaphor and antecedent match completely.

Head Match. Return whether the lowercased head strings of anaphor and antecedent match completely.

Alias. Return whether anaphor and antecedent are in an alias relation. This feature is only triggered if both mentions are proper names of the same type (either *Person*, *Organization* or *Location*) and have different heads. Depending on the type, different heuristics for assessing whether an alias relation between the two mention holds are employed. For organizations and locations, we check for abbreviations and whether one mention starts with the other. For persons, we run various name matching checks.

Tokens contained. Return whether all tokens of one mention are contained in the other mention (ignoring case).

[1] To convert we use PyStanfordDependencies, available at `https://github.com/dmcc/PyStanfordDependencies`. We use version 0.2.0.

Head contained. Return whether all head tokens of one mention are contained in the other mention's head (ignoring case).

Modifier. Return whether all pre- and post-modifiers of the anaphor are contained in the antecedent. We ignore case and discard all determiners, prepositions and apostrophe s. This features captures that mentions tend to be non-coreferent when they convey different information in the modifiers (Cai et al., 2011).

7.2.2.2 Distance Features

Features that measure the distance between two mentions are considered useful for pronoun resolution, since personal pronouns tend to refer to antecedents that are not too far away (Mitkov, 2002, p. 17f.).

Sentence Distance. Return the distance between anaphor and antecedent in sentences, capped at 5.

Token Distance. Return the distance between anaphor and antecedent in tokens, capped at 10.

7.2.2.3 Miscellaneous Features

Lastly, we consider features that do not fit into the categories above.

Embedded. Return whether one of the mentions embeds the other mention. This often rules out rules out coreference. Consider the following example:

(24) [[His]$_1$ friend]$_2$ is nice.

The mentions *His* and *His friend* cannot be coreferent[2].

Same Speaker. Return whether anaphor and antecedent have the same speaker. This feature relies on the speaker annotation in the OntoNotes corpus we evaluate on (Weischedel et al., 2013). With this feature, we can resolve coreference of first and second person pronouns in conversations, as mentioned in Section 2.1:

[2]This feature approximates the *i-within-i* constraint (Chomsky, 1981).

(25) A: How are [you]$_1$?

B: [I]$_1$'m fine, and you?

We can only resolve the coreference reliably if we know that *you* and *I* have different speakers.

7.3 Entity-based Features

Entity-based features differ from local features by their scope. While local features operate on pairs of mentions, entity-based features also take previous coreference decisions into account. For convenience, we define most entity-based features on hyperedges (X, Y), where X and Y are sets of mentions. If the latent structure of the approach in focus does not rely on hypergraphs, we obtain the sets X and Y by extracting the partial entity information encoded in the latent structure.

Some features examine relations between the sets X and Y, while other features only apply to one set of mentions. For the latter case, the features are extracted for both X and Y.

7.3.1 Features Adapted from Local Features

Following previous work on entity-based coreference resolution approaches (Luo et al., 2004; Yang et al., 2008; Rahman and Ng, 2011a), we induce many entity-based features from local features. Let us first discuss mention features. In Order to extend these to entity-based features, we extract the mention features for the pair $(m, n) \in X \times Y$ that minimizes the mention distance[3].

For the pairwise features, we employ two strategies. All pairwise features except for the distance features are binary. We apply logical predicates to these features. For a feature `fname`, we create four binary features `fname-ALL`, `fname-MAJ`, `fname-SOME`, `fname-NONE`. `fname-ALL` holds for a pair (X, Y) if `fname` holds for all pairs $(m, n) \in X \times Y$; `fname-MAJ` holds if the feature holds for the majority of pairs; `fname-SOME` holds if the feature holds for some, but not for the majority of pairs; `fname-NONE` holds if the feature is not true for any pair.

[3]We also experimented with other extraction schemes such as extracting features for all pairs $(m, n) \in X \times Y$, but the scheme described here gave best performance in preliminary experiments.

Distance features require a different handling. Following previous work (Yang et al., 2008), we handle distance features analogously to mention features: we compute the distance features for the pair $(m, n) \in X \times Y$ that minimizes the mention distance.

Following previous work (e.g. Webster and Curran (2014)), we also introduce the following feature that aggregates multiple mention features:

Agreement. Return whether all, the majority, some or none of the induced mention pairs agree in number, gender and semantic class. Two mentions agree in number/gender/semantic class if the corresponding value is *Unknown* for one mention, or if the values match.

7.3.2 Entity-based Features not Regarding the Structure

We now discuss features of hyperedges (X, Y) that go beyond aggregation of local features. We first discuss features that do not examine any structure of of X or Y (such as anaphor-antecedent relations). In this thesis, we want to keep the feature selection for different models simple and therefore only use structure-ignorant features that yielded consistent improvements in the literature. The only such feature is *cluster size*, which was used in most work on entity-centric models (Culotta et al., 2007; Björkelund and Kuhn, 2014; Webster and Curran, 2014; Clark and Manning, 2015, inter alia).

Cluster Size. Return the binned size of the partial entities (in number of mentions). The bins we use are 1, 2, 3, 4, 5, 6–10, 11–15, 16–20 and 21+.

7.3.3 Structural Entity-based Features

We experimented with a range of structural features, such as the in-degree of nodes, path lengths in antecedent trees and graphs, and existence of sibling nodes (whether there exists a mention that has the same antecedent as the mention in focus). However, we found none of these features to be effective in preliminary experiments. Only the *antecedent has antecedent* feature gave slight improvements. We therefore only use this feature in our experiments. In contrast to the other entity-based features, this feature is defined by the context of a pair of mentions in the latent structure. It is only applicable to tree/graph-based models.

Antecedent has Antecedent. Return whether the antecedent has a preceding node m_i in the tree/graph that is not the dummy mention m_0. Additionally return the fine-grained mention type of the closest such mention.

7.3.4 Discussion

We only use a small set of entity-based features that go beyond the aggregation of local features. Hence, we cannot expect large performance gains from the additional features. However, by only using these few features we can avoid complex feature selection and can still analyze the contribution of features that have proven to be well-working in the literature.

7.4 Feature Combinations

Our framework bases learning and prediction on a linear model. Since these models can not capture interactions between features, we have to provide the model with feature combinations to obtain competitive performance.

We employ two stages of feature combinations. In the first stage, we go through each mention feature, and concatenate the feature value of the anaphor with the feature value of the antecedent. For example, consider the pair of mentions (*him*, *the president*). For the *head* feature, we will extract three features:

- the head of the anaphor, $\text{head}_{\text{anaphor}}(\text{him}, \text{the president}) = \text{him}$,
- the head of the candidate antecedent,
  $\text{head}_{\text{antecedent}}(\text{him}, \text{the president}) = \text{president}$,
- the concatenation of the heads, $\text{head}_{\text{concat}}(\text{him}, \text{the president}) = \text{him+president}$

This enables the machine learning models to access further relational information between anaphor and antecedent.

In the second stage, we follow Durrett and Klein (2013) and concatenate each feature with all *fine-grained mention type* features. In the (*him*, *the president*) example, we would add concatenations of all features with each of the following three fine-grained mention type features:

- the fine-grained mention type of the anaphor,
  $\text{fine\_type}_{\text{anaphor}}(\text{him}, \text{the president}) = \text{he}$,
- the fine-grained mention type of the candidate antecedent,
  $\text{fine\_type}_{\text{antecedent}}(\text{him}, \text{the president}) = \text{Definite}$,
- the concatenation of the mention types,
  $\text{fine\_type}_{\text{concat}}(\text{him}, \text{the president}) = \text{he+Definite}$

This has the effect of learning type-specific weights at a fine-grained level. We extend the combination scheme to features of hyperedges (X, Y) by concatenating the fine-grained mention types of the first mention of X and the last mention of Y with the feature.

8 Experiments and Analysis

In Chapter 6 we demonstrated how approaches to coreference resolution can be expressed in our framework. By expressing the approaches according to a uniform representation, we were able to transparently analyze differences and similarities between the approaches in terms of their structure. Furthermore, we carved out parameters of each approach which we discussed in detail. We now complement this structural analysis by a quantitative and qualitative analysis on a benchmark corpus. For doing so, we will mainly employ the error analysis framework which we presented in Chapter 4.

We start by describing the experimental setup, including data sets, mention detection and evaluation (Section 8.1). We then evaluate, analyze and compare implementations of approaches based on different latent structures. We focus on an in-depth analysis of various mention pair models and ranking architectures (Sections 8.2 and 8.3), as even these fundamental coreference resolution approaches are not well understood. We additionally evaluate and analyze selected entity-based models (Section 8.4). We conclude our evaluation of the models by discussing their performance on unseen test data (Section 8.5). Finally, we give a summary of the analysis presented in this chapter (Section 8.6).

8.1 Experimental Setup

Before analyzing the approaches, we describe the experimental setup. This includes the data set, preprocessing, the feature set and the evaluation parameters.

8.1.1 Data Sets

We conduct all experiments and evaluation on the English portion of the CoNLL-2012 shared task data test (Pradhan et al., 2012), which is a subset of the OntoNotes 5.0 corpus (Weischedel et al., 2013). This data set is the standard for evaluating approaches

to English noun phrase coreference resolution (see, e.g., Björkelund and Kuhn, 2014; Durrett and Klein, 2013; Clark and Manning, 2015).

Set	Documents	Mentions	Links	Entities
Training	2,802	155,560	120,417	35,134
Development	343	19,156	14,610	4,546
Test	348	19,764	15,232	4,532

Table 8.1: Statistics about the CoNLL-2012 English shared task data set.

In this data set, all forms of coreference are annotated. In addition to noun phrase coreference, also event coreference is annotated. In this thesis, we do not attempt to solve any coreference relations involving event coreference.

Documents in the data set stem from seven genres, which are broadcast conversations, broadcast news, magazine texts, news wire, pivot texts, telephone conversations and web logs. The data set has a predefined training/development/test split, to which we adhere in all experiments. Table 8.1 summarizes the number of documents, mentions, links and entities in the data set. Note that according to the OntoNotes annotation guidelines *singleton mentions* – expressions which have a referent, but no coreferring mention – are not annotated.

8.1.2 Mention Detection

We employ a rule-based mention extractor, who extracts mentions from the parse and named entity annotation layers. The mention extractor was originally implemented by Samuel Broscheit (Martschat et al., 2012) and modeled after the mention extractor described in Lee et al. (2011). A refined reimplementation was conducted by the author of this thesis.

The OntoNotes coreference annotation guidelines for English state that "All noun phrases with distinct headwords are extracted [...] Whenever head-sharing NPs are nested, the largest logical span is used in co-reference" (Weischedel et al., 2013, Coreference Guidelines for English OntoNotes, p. 3). Hence, we first extract all noun phrases and all named entity chunks. We then extract heads via a modified version of the head rules presented in Collins (1999). If the head token has a named entity tag or if its part-of-speech tag starts with *NNP*, we recognize the mention as a proper name

and employ a heuristic to extract a more meaningful head. The head for a proper noun starts at the first token tagged as a noun until a punctuation, preposition or subclause is encountered. Coordinations have the *CC* tagged token as head.

Finally, we run the following set of filters:

1. If two mentions have the same head, discard the mention with the smaller span.
2. If the head of one mention is embedded in the head of another mention, discard the mention with the embedded head.
3. Discard any mention which is embedded in an apposition.
4. Discard all mentions whose head has the part-of-speech tag *JJ*.
5. Discard all mentions with surface form “mm”, “hmm”, “ahem”, “um”, “US” or “U.S.”.
6. Discard all mentions whose head is a proper name of type *Quantity*, *Cardinal*, *Ordinal*, *Money* or *Percent*.
7. Discard any “it” which appears in the context *it * * that* or *it * * * that* (such as *it is known that*).
8. Discard any “you” which appears in the context *you know*.

The first five filters filter mentions which are never annotated as coreferent according to the annotation guidelines[1]. The sixth filter discards proper names where the resolution is very unreliable. Finally, filters seven and eight are heuristics for recognizing pleonastic “it” and “you”.

8.1.3 Features

If not noted otherwise, all models employ the same feature set consisting of all local features described in Sections 7.2. The features are combined using the scheme described in Section 7.4. Entity-based models additionally employ entity-based features. We describe how we use these features in the corresponding sections.

[1] Per the annotation guidelines, nationality acronyms as pre-modifiers are not eligible for coreference (Weischedel et al., 2013, Co-reference Guidelines for English OntoNotes, p. 8).

8.1.4 Evaluation

We now describe parameters for evaluation. These include evaluation metrics, error analysis and hyperparameter optimization.

8.1.4.1 Evaluation Metrics

We score the output of each model with version v8.01 of the CoNLL scorer (Pradhan et al., 2014; Luo et al., 2014)[2]. The scorer outputs recall, precision and F_1 scores for MUC, B^3, $CEAF_m$, $CEAF_e$ and BLANC (for details on these metrics, see Section 2.4).

In the CoNLL shared tasks on coreference resolution (Pradhan et al., 2011, 2012), the average of MUC, B^3 and $CEAF_e$ F_1 score was adopted as the official evaluation metric for ranking the participants. This method of evaluation was subsequently adopted by many researchers (Durrett and Klein, 2013; Chang et al., 2013; Björkelund and Kuhn, 2014, inter alia). We follow this common practice and employ the average of MUC, B^3 and $CEAF_e$ F_1 score (also referred to as the *CoNLL score*) as our main metric for comparison.

We refrain from testing statistical significance of the differences in evaluation metrics on development data, since the models were tuned and analyzed on this data. When testing for statistical significance of the differences in evaluation metrics on test data, we employ an approximate randomization test (Noreen, 1989)[3].

8.1.4.2 Error Analysis

One aim of the research presented in this chapter is to compare approaches to coreference resolution based on the errors they make. By doing so, we can identify similarities and differences between approaches, as well as assess strengths and weaknesses of individual approaches. We then can connect these properties of the models to the structures and parameters they rely on.

Spanning Tree Algorithms. We perform the analysis based on the error analysis framework presented in Chapter 4. In this framework, we can choose between various algorithms for spanning tree computation, where each algorithm leads to a different notion of an error.

[2]Available for download at `http://conll.github.io/reference-coreference-scorers/`.
[3]The implementation we use is available at `http://smartschat.de/software`.

We will extract spanning trees for precision errors based on the pairwise output of the approaches (if available), while we will extract spanning trees for recall errors based on pairwise scores (if available). We employ this spanning tree algorithm for precision errors since the pairwise output provides a natural interpretation as a spanning tree for system output. If the pairwise output of the approach does not constitute a spanning tree, we fall back to the spanning tree algorithm based on pairwise scores. If there are no pairwise scores available, we fall back to computing spanning trees based on accessibility. On the other hand, we employ the score-based algorithm for recall errors, since it will create spanning trees that consist of links that are considered "easier" than other links by the system under consideration. If there are no pairwise scores available, we fall back to computing spanning trees based on accessibility.

Putting the Error Numbers into Context. In order to put the error counts into context, we compare the numbers with *upper bounds* and with *resolved links*.

Let us first discuss the upper bounds. We want to bound the number of recall errors by the maximum amount of recall errors the system can make, and the number of precision errors by the maximum amount of precision errors the system can make. For both recall and precision errors, an upper bound for the number of errors is the number of edges in the respective spanning trees. Both of these numbers have an intuitive interpretation: The number of edges in reference spanning trees corresponds to the number of recall errors made by a coreference system that puts each mention into its own coreference chain. The number of edges in system spanning trees corresponds to the number of precision errors the system would make if every attempted anaphor-antecedent decision was incorrect. Observe that the upper bound for recall errors is the same for each system, while the upper bound for precision errors depends on the output of the system. For example, if a mention ranking system makes 12,000 anaphor-antecedent predictions, the upper bound for precision errors is 12,000. If it only makes 11,000 anaphor-antecedent predictions, the upper bound is 11,000.

For the resolved links, we take each edge from the respective spanning tree. When considering recall errors, we check whether the mentions are in the same system entity. When considering precision errors, we check whether the mentions are coreferent according to the reference annotation.

Error Categorization. To understand the errors made by a system, we will compare the distribution of the errors and of the resolved links according to various linguistic

criteria. Each error in our framework is represented by a link (m_j, m_i) between two mentions, where m_j appears after m_i. We call m_j the *error anaphor* and m_i the *error antecedent*. When there is no danger of ambiguity in context, we just call m_j the anaphor and m_i the antecedent. From the discussion of the linguistics of coreference in Section 2.1 we saw that different classes of anaphors (for instance proper names and pronouns) differ in how they should be treated: they exhibit different behavior in terms of reference, appear in different contexts and require different knowledge sources to establish the coreference relations. Similar observations hold for different classes of antecedents.

Hence, we categorize the errors with respect to the types of anaphor and antecedent. Since we observed that proper name/common noun coreference and pronoun coreference behave differently, we distinguish between links where the anaphor is a personal pronoun, where both mentions are proper names or common nouns, and all remaining cases. Errors where the anaphor is a personal pronoun are further distinguished by the canonical form of the pronoun (*I*, *you*, *we*, *he*, *she*, *it* and *they*). When presenting the numbers, to avoid clutter, we summarize numbers for first/second person pronouns, third person gendered pronouns and third person ungendered pronouns. Name/noun errors are further distinguished by the types of the mentions (*Both name*, *Mixed* and *Both noun*). We do not further distinguish the remaining errors. We call the category containing all remaining errors *Misc*.

8.1.4.3 Hyperparameter Optimization

The performance of the approaches which we will compare depends on various hyperparameters, such as the exact form of the cost function and the number of iterations the model is trained. To allow for a fair comparison, it is important that we run each approach with its optimal set of hyperparameters.

To obtain these optimal hyperparameters, we perform a grid search for optimal parameters by training on the training data and evaluating on the development data. The configuration which led to the highest CoNLL score is then fixed for future experiments. The only hyperparameter handled differently is the number of iterations of the perceptron algorithm: for each model, we train the model for 50 iterations, and pick the iteration whose model yielded the highest result on the development data.

8.2 Mention Pair Models

We now evaluate all structures discussed in Chapter 6, beginning with the mention pair model (Soon et al., 2001). As we have discussed in Chapter 6, there are many variants of the mention pair model. While the structure they are based on is shared by most approaches, there are substantial differences between individual implementations. We now discuss and analyze these differences by varying parameters of the mention pair model, including the handling of the training data, the method to obtain coreference information from the predicted latent structures, and the factorization into substructures.

8.2.1 A Vanilla Mention Pair Model

We want to start our evaluation and analysis with an implementation that makes as few modeling assumptions as possible. In particular, we will not use any heuristics for changing the distribution of the training data. Furthermore, following models from the literature, we consider each edge of the latent structure (which is a pair of mentions) as a substructure.

The only remaining parameter is the function to obtain coreference information from the latent structure. Since we learn from all mention pairs in the data, in particular from all coreferent pairs, we opt to employ *aggressive merge* clustering, since it is the clustering algorithm that aligns best with the latent structures considered during training.

We dub the resulting model the *vanilla mention pair model*. This simple model will be our baseline and will serve as a starting point for further analysis.

8.2.1.1 Results

The model only has one hyperparameter, which is the number of perceptron iterations. We optimized this parameter according to the procedure described in Section 8.1.4.3.

Table 8.2 displays the result of the model on CoNLL'12 English development data. To put the results into context, we compare with *nn_coref*, the state-of-the-art neural network system by Wiseman et al. (2015)[4] and with *Stanford Sieve* (Lee et al., 2013),

[4]Available for download at `https://github.com/swiseman/nn_coref`. During writing of this thesis, *nn_coref* was the best-performing coreference resolution system. Subsequently, the state-of-the-art has been improved by Wiseman et al. (2016) and Clark and Manning (2016). The performance of

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
nn_coref	69.82	75.91	72.74	57.89	66.21	61.77	56.74	60.65	58.63	64.38
Stanford Sieve	65.91	64.09	64.99	58.66	50.91	54.51	48.58	54.27	51.27	56.92
Vanilla pair	65.77	76.90	70.90	55.19	53.78	54.48	36.09	64.07	46.17	57.18

Table 8.2: Results of the vanilla mention pair model on CoNLL'12 English development data, compared with *nn_coref* (Wiseman et al., 2015) and *Stanford Sieve* (Lee et al., 2013).

the recent version of the system that won the CoNLL-2011 shared task on coreference resolution (Pradhan et al., 2011; Lee et al., 2011)[5]. *Stanford Sieve* is among the most widely-used coreference resolution systems. While there is a gap of more than 7 points average F_1 to the state-of-the-art system, our model performs favorably compared to *Stanford Sieve*.

8.2.1.2 Error Analysis

To better understand strengths and weaknesses of the vanilla mention pair model, we perform an error analysis on the output of the model, as described in Section 8.1.4.2. For determining the coreference chains, the model outputs all pairs that it considers coreferent, therefore the output does not necessarily constitute a spanning tree. We therefore employ the spanning tree algorithm based on pairwise scores to compute precision errors.

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Vanilla Pair	5,000	14,609	34%	3,803	12,495	30%

Table 8.3: Overview of recall and precision errors of the vanilla pair model.

Table 8.3 summarizes all recall and precision errors made by the vanilla mention pair

Clark and Manning (2016) in terms of average F_1 is 66.01 on development data and 65.29 on test data.

[5]Available for download at `http://stanfordnlp.github.io/CoreNLP/`. We use version 3.6.0 and run the included *deterministic* coreference resolution system.

model. The upper bounds are in the columns titled *Max*. Both for recall and precision, the model makes roughly one third of the maximum errors. We now investigate both recall and precision errors in detail.

Recall Errors. The top of Figure 8.1 gives an overview of the recall errors, categorized by the mention type of the participating mentions. The figure compares the resolved links (in blue) with the errors (in the remaining colors). The errors are further divided into[6]

- errors due to not resolving a mention at all (red);
- errors due to choosing a wrong antecedent for a mention (gray);
- errors that are due to mention extraction: at least one of the two mentions in the pair was not extracted by the mention extractor, therefore we can not successfully resolve the link (orange).

As we can see from the figure, for proper name pairs, first and second person pronouns, and gendered third person pronouns there are many more resolved links than errors. For the other categories, there are as many or more errors than links. Especially the *Mixed* and *Misc* categories have only few resolved links, which indicates that pairs from these categories are particularly difficult to resolve.

The impact of mention detection is more pronounced in the non-pronominal errors. Here the errors are mainly due to preprocessing errors or due to annotation inconsistencies. The high number of mention detection errors in the *Misc* category can be explained by the fact that we do not extract any verb phrases as mentions, since we do not attempt to resolve event coreference. However, there are roughly 200 verb phrases annotated as mentions in the data.

For the remaining errors, which we can potentially resolve without changing the mention detection, almost all errors can be attributed to not attempting to resolve the mention at all. We hypothesize that this is due to the resolution strategy of the vanilla mention pair model: since it follows an *aggressive merge* approach, a mention can have many antecedents, which renders it less likely that we miss a link between two mentions that belong to the same coreference chain. We will investigate this

[6]This division and the division of precision errors are similar to the classes used by Durrett and Klein (2013) in their cost function for a mention ranking model.

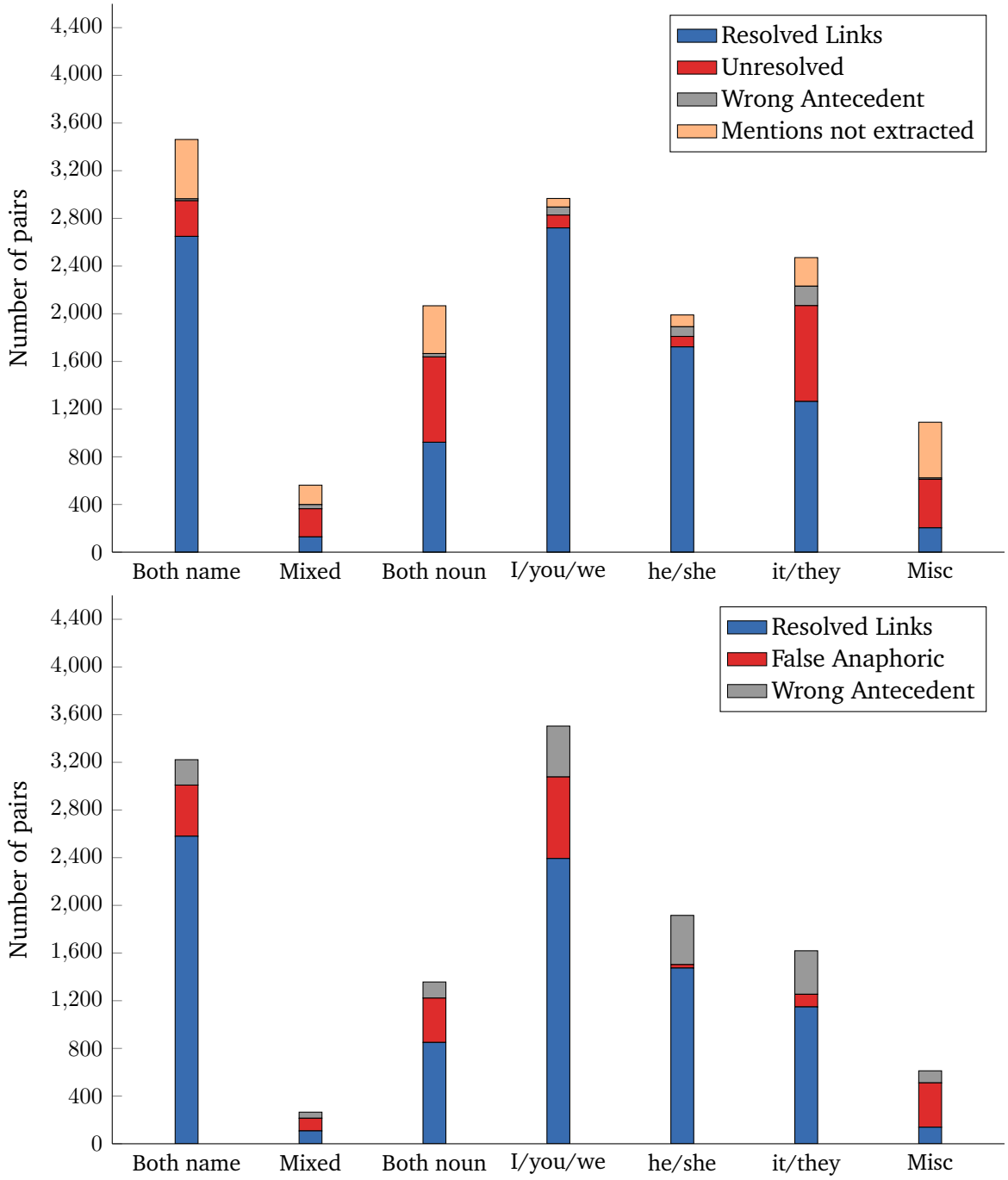

Figure 8.1: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of the vanilla mention pair model on CoNLL-2012 English development data.

hypothesis when we compare *aggressive merge* with other clustering schemes in the next subsection.

Qualitatively, for most name/noun recall errors there is string similarity. For instance, for the proper name pair errors where both mentions were identified by the mention detection, there is token overlap (ignoring determiners) in 79% of the cases (but only in 10% the strings match completely). Similar numbers hold for the common noun pair errors.

Precision Errors. The bottom of Figure 8.1 gives an overview of all categorized precision errors. Again, resolved links (in blue) are compared with errors (in different colors). For precision errors, we divide the errors into

- errors due to choosing an antecedent for a non-anaphoric mention (red);
- errors due to choosing a wrong antecedent for an anaphoric mention[7] (gray).

Recall that the vanilla mention pair model uses aggressive merge clustering, and therefore a mention can receive multiple antecedents. Therefore, although a mention has a wrong antecedent among its antecedents, it can also have a correct antecedent among its antecedents. If this is the case, there is no recall error for this mention.

For all categories except *Mixed* and *Misc* there are more resolved links than errors. Most pronoun resolution errors are caused by choosing wrong antecedents for anaphoric pronouns. For name/noun errors, the picture is reversed. The low number of *False Anaphoric* errors for *he/she* can be explained by the fact that of all occurrences of these pronouns in the corpus, 98% are anaphoric. For *it/they*, the model seems to learn a cautious approach: although 37% of all occurrences of *it* are non-anaphoric (roughly 600 cases), there are only comparatively few errors due to choosing an antecedent for such mentions.

Qualitatively, many precision errors exhibit string similarity. Table 8.4 shows for each category the proportion of precisions errors where the heads match. The numbers are particularly high for name/noun errors, but also for almost one third of pronoun resolution precision errors the heads match.

[7] These do not correspond exactly to the similar error class for recall errors, since a correct antecedent may not be in the set of extracted mentions. Therefore, the error counts as a mention extraction error when computing recall errors. For precision errors, we do not perform a further division of this error class into errors due to mention extraction and not due to mention extraction, since we want to examine the factors that lead the approaches to choosing erroneous antecedents. We do not aim to analyze the impact of mention detection in detail.

	Both name	Mixed	Both noun	I/you/we	he/she	it/they	Misc
% matching heads	72%	20%	94%	50%	32%	34%	3%

Table 8.4: Proportion of precision errors with matching heads.

8.2.1.3 Discussion

The vanilla mention pair model is easy to describe and implement. While not being state-of-the-art, the performance is higher than the performance of the recent version of Stanford Sieve (Lee et al., 2013). An error analysis reveals that the main sources of errors are mention extraction errors and issues with anaphoricity detection (*Unresolved* and *False Anaphoric* errors). Choosing wrong antecedents for mentions has a substantial effect mainly for pronouns.

8.2.2 Variations in the `obtain_coreference` Function

We will now consider how more sophisticated variants of the mention pair model tackle the issues of the vanilla mention pair model. As we saw in the chapter about related work (Chapter 3), a large portion of the work on the mention pair model was concerned with improving the algorithm to obtain coreference chains from the classification output of the mention pair model. In our framework, these algorithms correspond to instantiations of the `obtain_coreference` method.

We consider the two instantiations beyond *Aggressive Merge* described in Section 6.2.1: *Closest First* and *Best First*.

8.2.2.1 Results

Table 8.5 shows the results of the mention pair model with different instantiations of the `obtain_coreference` function as discussed above. Both clustering methods, closest first and best first, improve in average F_1 over the vanilla pair baseline, which employs aggressive merge clustering. However, the vanilla pair model is the best-performing model according to the MUC metric. The gains over the vanilla pair model are due to improved precision for MUC and B^3, and improved recall for $CEAF_e$. Best-first clustering performs slightly better (around 0.3 points average F_1) than closest-first clustering.

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Vanilla pair	**65.77**	76.90	**70.90**	**55.19**	53.78	54.48	36.09	**64.07**	46.17	57.18
Closest First	61.19	**78.36**	68.72	45.88	**67.19**	54.53	**42.57**	59.79	**49.73**	57.66
Best First	61.47	78.27	68.86	47.62	66.95	**55.65**	41.47	61.05	49.39	**57.97**

Table 8.5: Results of mention pair variants with different `obtain_coreference` instantiations. Highest values for each column are marked bold.

8.2.2.2 Error Analysis

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Vanilla Pair	5,000	14,609	34%	3,803	12,495	30%
Closest First	5,668	14,609	39%	2,936	11,410	26%
Best First	5,628	14,609	39%	3,016	11,474	26%

Table 8.6: Overview of recall and precision errors of mention pair models with different `obtain_coreference` instantiations.

Since the models using closest-first clustering and best-first clustering output spanning trees of system entities, we employ the spanning tree algorithm based on system output for extracting precision errors. For extracting recall errors, we employ the spanning tree algorithm based on pairwise scores. Unless noted otherwise, we will from now on employ these spanning tree algorithms for all analyses conducted in this chapter.

Table 8.6 and Figure 8.2 summarize all recall and precision errors of the model variants and compare the numbers with the errors made by the vanilla pair model. We can see a substantial reduction in precision errors, while recall errors increase. We now compare the errors in detail.

Closest First vs. Vanilla Pair. For recall errors, we observe substantial differences for the categories *I/you/we* and *he/she*. Compared to the vanilla pair baseline, there is an increase in the number of recall errors (in particular *Wrong Antecedent* errors) and

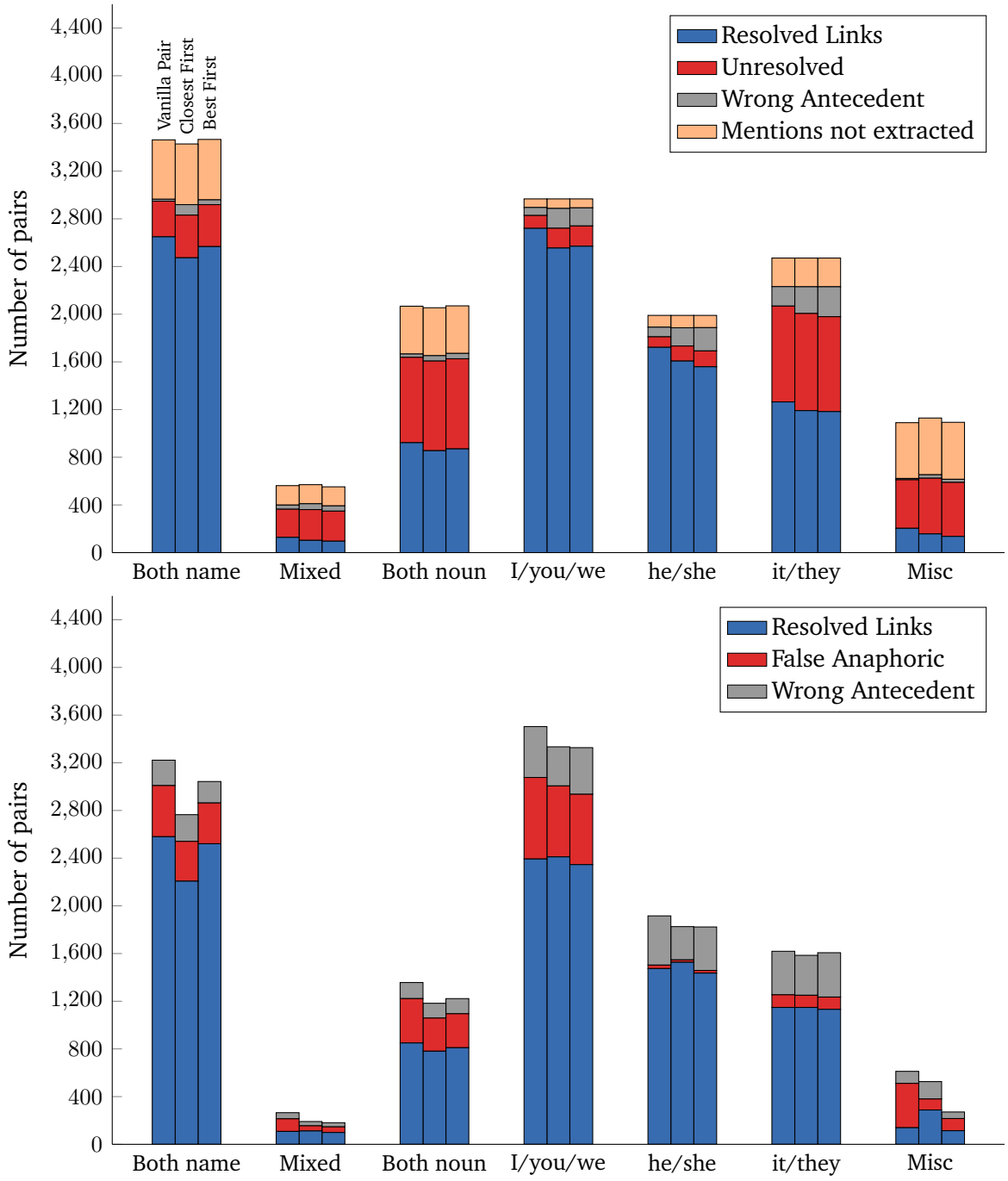

Figure 8.2: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of the mention pair variants on CoNLL-2012 English development data. Left bar: vanilla mention pair model; middle bar: model employing closest-first clustering; right bar: model employing best-first clustering.

decrease in the number of resolved links for for these categories. Similar differences for most other categories are less pronounced, but also visible. Regarding precision errors, for most categories there is a slight increase in resolved links and a reduction of errors. For the *Both name* and *Both noun* categories, resolved links as well as errors decrease.

The vanilla pair model obtains coreference chains by computing the transitive closure over all mention pairs that received a "+" label. The closest-first model only considers one "+"-labeled pair for each anaphor, which is the pair with the closest antecedent such that the label is "+". The vanilla pair model, employing aggressive merge clustering, selects many antecedents for mentions, including many wrong antecedents. This leads to many *Wrong Antecedent* precision errors and, since this method assigns many non-anaphoric mentions to coreference chains, also to many *False Anaphoric* errors for these mentions. Especially for pronoun resolution, the more cautious closest-first approach leads to a reduction in such errors.

Best First vs. Closest First. For recall errors, there are only slight differences. In general, the trends observed when switching from vanilla pair to closest first continue. One exception is the *Both name* category, where the number of resolved links slightly increases when applying best-first clustering. The differences are more pronounced for precision errors. Replacing closest-first clustering with best-first clustering usually yields a reduction in resolved links and an increase in errors. In particular, the number of *Wrong Antecedent* errors increases for pronoun resolution. For *Both name*, however, resolved links increase substantially. For *Misc*, there are less resolved links but also less errors.

While closest-first clustering selects the closest antecedent classified as coreferent, best-first clusterings selects the highest-scoring antecedent classified as coreferent[8]. For pronoun resolution, closest-first clustering yields better results. Hence, at least for the set of distance features we employ, the mention pair model is not able to learn parameters which accurately take distance into account when deciding between high-scoring candidate antecedents for pronouns. Best-first clustering changes the distribution of the output for non-pronominal anaphors. For proper names, for example, proper name antecedents tend to score higher, even though they might be farther away than candidate antecedents which have other mention types. This increases the number of resolved links in the *Both name* category, and decreases errors and resolved

[8]Where the score of an antecedent is the score of the corresponding pair.

links for the *Mixed* and *Misc* categories.

8.2.2.3 Discussion

Using closest-first or best-first clustering as the `obtain_coreference` function instead of aggressive-merge leads to improved performance, mainly due to a reduction in precision errors. In particular, the clustering algorithms that only choose one antecedent make less *Wrong Antecedent* precision errors for pronoun resolution. Compared to closest-first clustering, best-first clustering leads to decreased performance regarding pronoun resolution, but improves the resolution of proper names.

8.2.3 Resampling Heuristics

Soon et al. (2001) resample the training data to obtain training instances that align with their clustering approach: for each anaphoric mention m_j, they only retain the closest preceding coreferent mention m_i and discard all mentions appearing before m_i. Besides aligning with the clustering approach, this procedure has also the effect of improving the balancing of the number of positive and negative training instances.

We now evaluate and analyze the contribution of such resampling schemes. In particular, we consider

- the scheme by Soon et al. (2001): for a mention m_j, let m_i, $i > 0$, be the closest preceding mention such that the edge (m_j, m_i) has label "+". If no such mention exists, discard all edges (m_j, m_k) with $k < j$. Otherwise, discard all edges (m_j, m_k) with $k < i$;
- a modified Soon et al. (2001) scheme. The scheme described above is very restrictive: each mention is required to have an antecedent and the model learns only from anaphoric mentions. We relax this requirement slightly and consider all mentions as sources for edges that are in some coreference chain (the Soon et al. (2001) scheme does not consider the first mention in the chain). The first mention in a chain is not anaphoric, hence the model also learns from non-anaphoric mentions. Formally, for a mention m_j, let m_i be the closest preceding mention such that the edge (m_j, m_i) has label "+". If $i = 0$ and m_j has no incoming edge with label "+"[9], discard all edges (m_j, m_k) with $k < j$. Otherwise, discard all edges (m_j, m_k) with $k < i$;

[9] This is equivalent to m_j being in no coreference chain.

We also experimented with further variants, such as inducing negative examples by *all* preceding non-coreferent mentions m_i of a mention m_j, but we could not observe improved performance with these variants. We do not discuss these variants further in this thesis.

8.2.3.1 Results

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Vanilla pair										
All pairs	65.77	76.90	**70.90**	55.19	53.78	54.48	36.09	**64.07**	46.17	57.18
Soon	**79.42**	47.14	59.16	**73.86**	20.20	31.73	26.24	36.49	30.53	40.47
Mod. Soon	71.66	69.52	70.57	62.56	49.54	55.29	45.89	55.63	50.30	58.72
Closest first										
All pairs	61.19	**78.36**	68.72	45.88	**67.19**	54.53	42.57	59.79	49.73	57.66
Soon	73.23	50.19	59.56	62.62	35.90	45.64	48.24	40.48	44.02	49.74
Mod. Soon	68.13	71.34	69.70	54.50	61.45	57.77	**54.85**	54.27	**54.56**	**60.68**
Best first										
All pairs	61.47	78.27	68.86	47.62	66.95	55.65	41.47	61.05	49.39	57.97
Soon	71.95	49.31	58.52	62.19	35.22	44.97	44.57	39.38	41.82	48.44
Mod. Soon	66.87	72.38	69.52	53.86	63.30	**58.20**	53.01	54.93	53.95	60.56

Table 8.7: Results of mention pair models with different resampling variants. Highest values for each column are marked bold.

Table 8.7 shows the results of the mention pair model with different variants for resampling and for the `obtain_coreference` function. Employing the resampling scheme as described by Soon et al. (2001) leads to a large increase in recall, but an even larger drop in precision for all `obtain_coreference` variants.

Employing our modified scheme, however, leads to increases for all metrics and `obtain_coreference` variants. While the scheme was developed to align with closest-first clustering, we can also observe an increase for aggressive-merge clustering (in the vanilla pair model) and for best-first clustering. Nevertheless, the increase is most pronounced for closest-first clustering.

8.2.3.2 Error Analysis

We analyze the impact of the resampling scheme exemplarily for the variant that employs closest-first clustering.

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
All pairs	5,668	14,609	39%	2,936	11,410	26%
Soon	3,910	14,609	28%	11,471	21,316	54%
Mod. Soon	4,654	14,609	32%	4,494	13,954	32%

Table 8.8: Overview of recall and precision errors of mention pair models employing closest-first clustering and different resampling variants.

Table 8.8 gives an overview of the recall and precision errors made by the the model using closest-first clustering and different resampling methods. Using resampling heuristics leads to a decrease in recall errors but an increase in precision errors. Most notably, when using the original Soon et al. (2001) scheme, the number of anaphor-antecedent decisions almost doubles and the number of precision errors increases by a factor of four. Figure 8.3 compares the errors in detail.

Soon vs. All Pairs. Across all categories, there is a decrease in recall errors when employing the Soon et al. (2001) resampling scheme, in particular for common noun pairs and third-person ungendered pronouns. The majority of the reduced errors can be attributed to the successful resolution of mentions that were unresolved in the original model. The reduction in recall errors is accompanied by an increase in precision errors. In general, there are more errors due to resolving non-anaphoric mentions. The differences are most striking for the *Mixed*, *Both noun* and *Misc* categories, and to a lesser extent for the *it/they* category.

Let us now analyze these errors further. For over 94% of the *False Anaphoric* precision errors made by the model using the Soon et al. (2001) resampling scheme, there is a head match between the mentions. Arguably, since the model learns only from pairs (m_j, m_i) where m_j is anaphoric, string matching will provide a strong clue for coreference. Learning from all pairs avoids many of these wrong assignments, at the expense of lower recall.

The high number of precision errors for the *Mixed*, *Both noun* and *Misc* categories

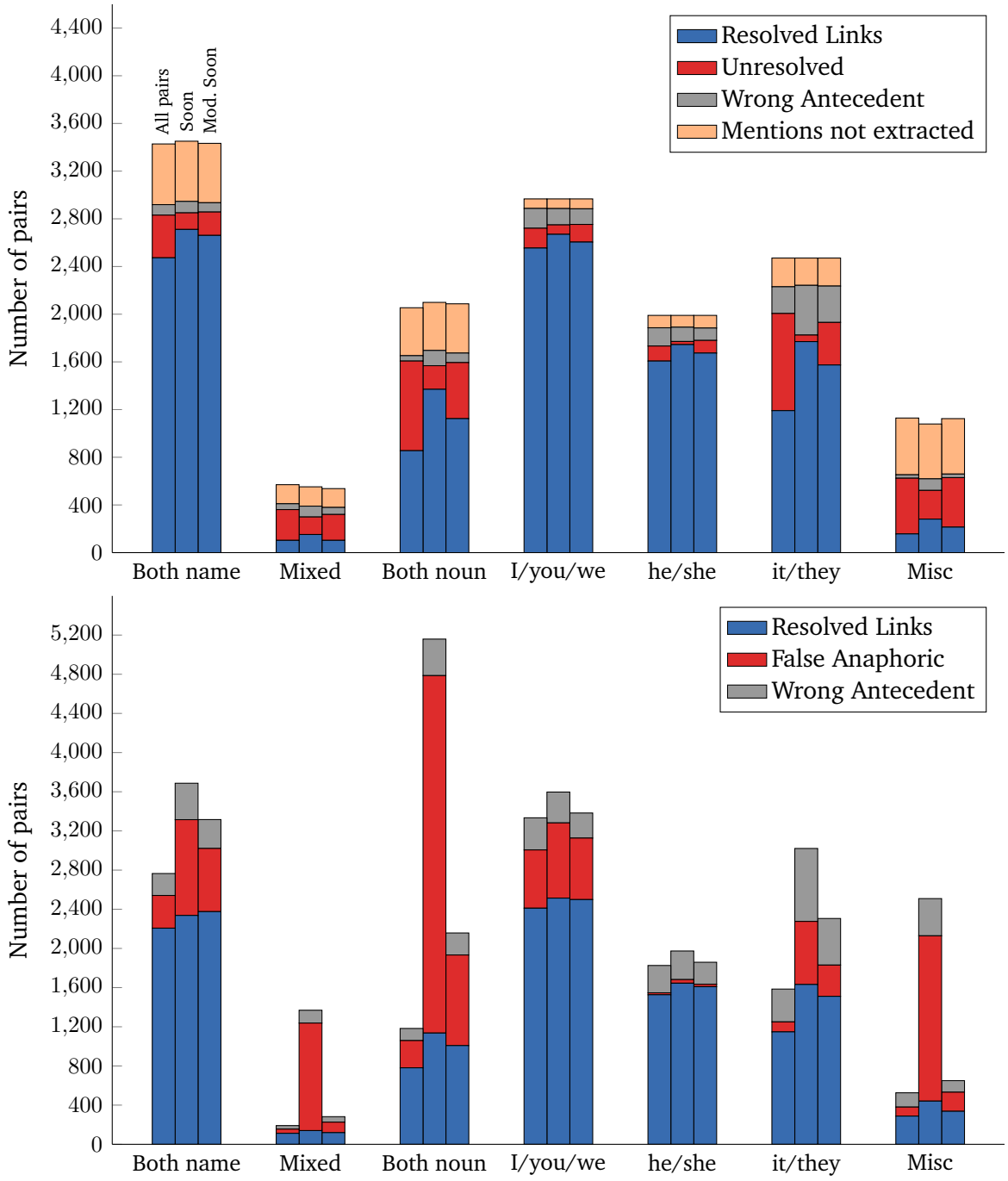

Figure 8.3: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of the mention pair variants with closest-first clustering and different resampling schemes on CoNLL-2012 English development data. Left bar: no resampling; middle bar: Soon et al. (2001) resampling; right bar: modified Soon resampling.

when employing the Soon resampling scheme can be explained similarly. Most mentions in these categories are not anaphoric. However, the model learns from data that suggests that every mention is anaphoric. Hence it finds antecedents for many of the non-anaphoric mentions in these categories.

Modified Soon vs. Soon. When switching from Soon et al. (2001) resampling to the modified Soon resampling scheme, there is an increase in recall errors due to more *Unresolved* errors. However, the number of recall errors is still much lower than the number of recall errors of the model that does not use resampling. There is a substantial reduction in precision errors. The differences can be explained by the fact that the modified Soon resampling scheme also considers mentions m_j that are first in their respective entities as sources of edges (m_j, m_i) when constructing graphs during training. Hence, the parameters are not solely learned from pairs (m_j, m_i) where m_j is anaphoric. This different resampling scheme leads to fewer *False Anaphoric* and slightly more *Unresolved* errors, since the model also considers mentions that do not have an antecedent. However, since the majority of mentions m_j in the edges (m_j, m_i) are anaphoric, the system still is biased towards predicting mentions as anaphoric.

8.2.3.3 Discussion

Both resampling schemes improve recall and lower precision for all `obtain_coreference` variants and all evaluation metrics. However, the drop in precision for the original Soon et al. (2001) resampling scheme compared to no resampling is too severe, which results in decreased overall performance. Employing the modified Soon et al. (2001) scheme improves overall performance.

From the error analysis we saw that the differences can mainly be attributed to differences in *anaphoricity determination*. Since the models using resampling schemes are mainly exposed to pairs (m_j, m_i) where m_j is anaphoric during training, they tend to propose more mentions as anaphoric by resolving them to an antecedent, which results in less recall errors but more precision errors. By design of the resampling scheme, this effect is much more severe for the Soon resampling scheme than for the modified Soon resampling scheme.

8.2.4 Substructures

Mention pair models from the literature consider each edge in the graph as an individual instance to learn from and to predict. In our framework, this paradigm corresponds to considering each edge of the latent structure as a substructure. We now study the effect of this factorization on the performance of mention pair models by investigating two other factorizations into substructures. Let $h = (M_x^0, A, L_A)$ be a latent structure for the mention pair model.

In the first variant, *Per Anaphor*, all mention pairs that share the same anaphor are considered as a substructure. Formally, this variant has the graphs

$$h_j = (\{m_0, \ldots, m_j\}, A_j, L_{A_j}) \tag{8.1}$$

with $A_j = \{(m_j, m_i) \mid j > i\}$ and $L_{A_j}(a) = L_A(a)$ as substructures. In the second variant, *Per document*, there is no factorization: all mention pairs for the whole document are considered.

We build both variants upon our best-performing mention pair model, which is the model employing the modified Soon scheme and closest-first clustering.

8.2.4.1 Results

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Baseline	**68.13**	71.34	**69.70**	**54.50**	61.45	57.77	54.85	54.27	**54.56**	**60.68**
Per Anaphor	66.96	72.17	69.47	53.04	63.46	**57.78**	**55.04**	53.60	54.31	60.52
Per Document	63.44	**76.30**	69.28	48.57	**68.67**	56.90	50.04	**55.64**	52.69	59.62

Table 8.9: Results of mention pair models with substructure factorization variants. The baseline considers each pair as a substructure and employs closest-first clustering and the modified Soon resampling scheme. Highest values for each column are marked bold.

Table 8.9 shows the results. Regarding the graph induced by each anaphor as a substructure leads to higher precision for MUC and B^3, but lowers recall. For $CEAF_e$ we observe the reverse effect. Overall, performance is slightly decreased. Regarding the

whole graph for each document as a substructure leads to a substantial increase in precision and a substantial drop in recall. Compared to the baseline, overall performance decreases by roughly 1 point.

8.2.4.2 Error Analysis

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Baseline	4,654	14,609	32%	4,494	13,954	32%
Per Anaphor	4,825	14,609	33%	4,207	13,556	31%
Per Document	5,340	14,609	37%	3,192	12,147	26%

Table 8.10: Overview of recall and precision errors when different substructure factorizations are employed.

Table 8.10 and Figure 8.4 summarize all errors made by the substructure factorization variants. As we can see, employing a factorization different from the baseline leads to less anaphor-antecedent decisions, reduces precision errors and increases recall errors. The effect is much more pronounced when employing a per document factorization. The most affected categories are *Mixed*, *Both noun* and *Misc*. In particular, *False Anaphoric* precision errors are reduced.

The analyzed models differ only with respect to the substructure factorization. Since all models use the same decoder which labels edges with the highest-scoring label, the factorization only affects training. Considering larger substructures delays updates of the parameter vector to the point when all edges that form the substructure are labeled. Judging from the distribution of errors, this strategy leads to a more cautious anaphoricity detection: we observe a decrease of *False Anaphoric* precision errors but an increase of *Unresolved* recall errors.

8.2.4.3 Discussion

Varying the factorization into substructures only affects training of the mention pair models. It mostly affects anaphoricity detection, leading to a more cautious approach with lower overall performance.

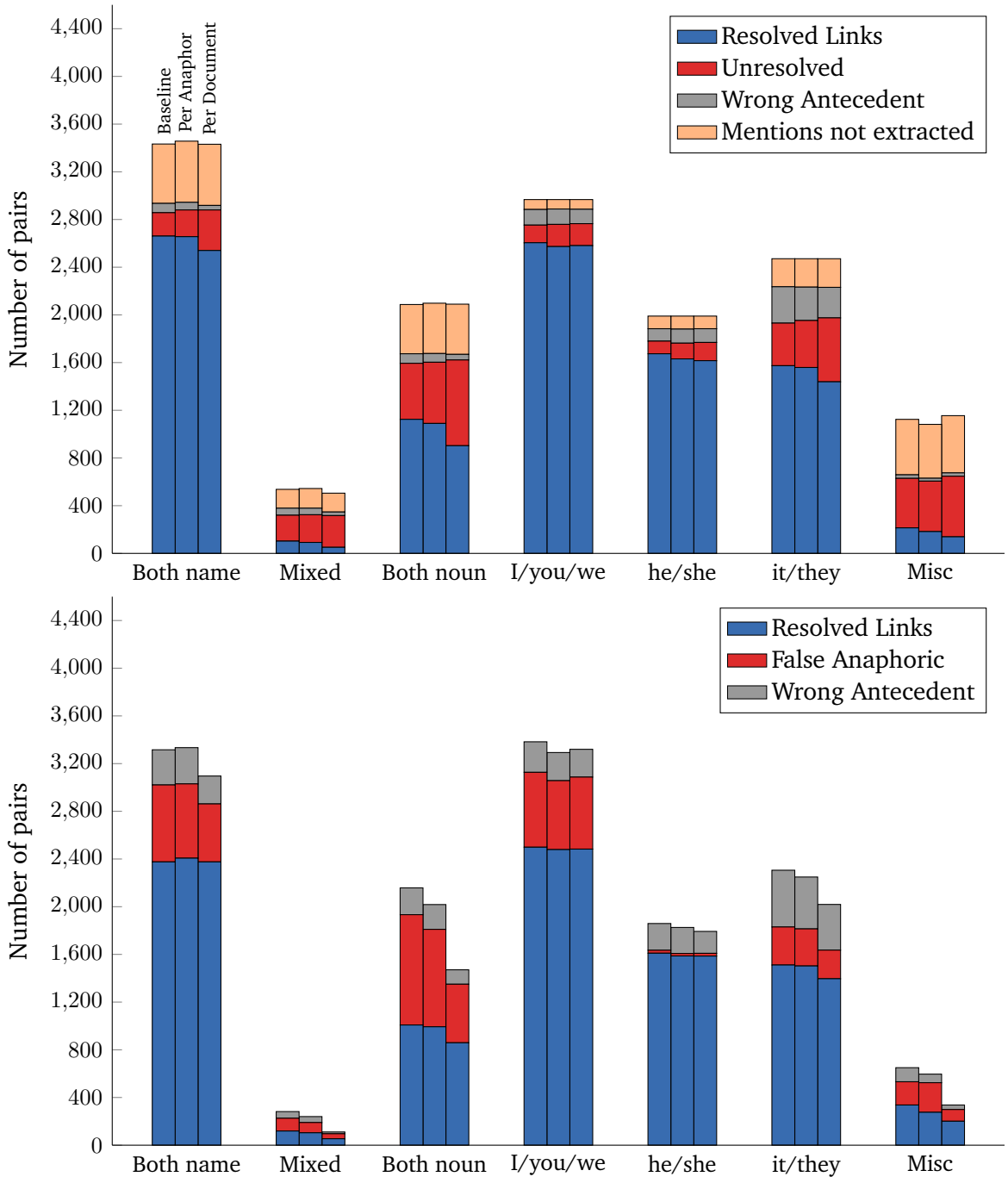

Figure 8.4: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of mention pair models with closest-first clustering, modified Soon resampling and different substructure factorizations on CoNLL-2012 English development data. Left bar: factorization into edges; middle bar: factorization into graphs induced per anaphor; right bar: no factorization.

8.2.5 Summary

Mention pair approaches model coreference resolution as binary classification of mention pairs. This perspective necessitates a clustering step during post-processing. We found that clustering strategies that choose at most one antecedent for each mention yield the best performance. Further performance boosts can be achieved by resampling the training data, in particular by using a novel resampling scheme based on the approach of Soon et al. (2001). While single-antecedent clustering schemes improve antecedent selection, the choice of the resampling scheme mainly affects anaphoricity detection. Using other substructures than mention pairs did not result in improved performance.

8.3 Mention Ranking and Antecedent Trees

The mention pair model is conceptually simple: pairs of mentions are labeled as either coreferent or non-coreferent. However, to obtain competitive performance, we need to apply heuristic resampling schemes and clustering algorithms.

Mention ranking (Denis and Baldridge, 2008; Chang et al., 2012) and antecedent tree (Fernandes et al., 2014) models, which we discussed in Section 6.3, are based on more sophisticated structures. As we will see, using these structures enables us to obtain competitive performance without using any resampling techniques or clustering algorithms.

8.3.1 A Simple Ranking Approach

To compare mention ranking with mention pair models, we first consider an instance of the mention ranking approach that is conceptually as similar as possible to a well-performing mention pair model. In particular, we want to expose the ranking model to the same data as the corresponding mention pair model. This will allow us to investigate the effect of switching from a mention pair architecture to a ranking architecture. However, we have to keep in mind that the data selection was optimized for a mention pair architecture.

The decoder for the mention ranking model always selects the highest-scoring edge for a substructure, which resembles best-first clustering. We therefore choose the mention pair model with the modified Soon resampling scheme and best-first clustering to

compare to.

The general mention ranking model discussed in Section 6.3 has the following degrees of freedom:

- pruning of graphs during training,
- the substructure-constraining function `constrain`, and
- the cost function c.

When using the modified Soon resampling scheme, the mention pair model learns only from mentions that are in coreference chains. Furthermore, during training it only considers antecedents up to the closest correct antecedent. When applying this to the mention ranking model, note that the mention ranking model by construction always decides on an antecedent for a mention. This is in contrast to the mention pair model, which may label all pairs for one anaphor as non-coreferent.

For the ranking model, we therefore apply the modified Soon resampling scheme to training as follows: we prune all substructures where the mention to compute the antecedent for is not in any coreference chain. For any retained substructure, let m_j be the mention to compute the antecedent for. Let m_i be the closest antecedent of m_j. We prune all edges (m_j, m_k) with $k < i$ and $k \neq 0$. We retain the edge (m_j, m_0) since this enables the model to decide whether a mention is anaphoric or not. In contrast to the mention pair model, the mention ranking model can not decide on this implicitly. The resulting model is very similar to the architecture proposed by Denis and Baldridge (2008), except that Denis and Baldridge (2008) employ the original Soon resampling scheme and use an anaphoricity classifier during preprocessing.

We also consider a second variant, *All Antecedents*, which allows the ranking model to consider all antecedents for a mention, which is closer to the setting when the model has to predict antecedents on unseen data[10]. For this variant, we prune all substructures where the mention to compute the antecedent for is not in any coreference chain, but do not prune any edges in the retained substructures. To enforce the model to only learn from the closest antecedent of a mention, we apply the substructure-constraining function `constrain` described in Section 6.3.1.

Since both variants change the distribution of instances in the training data, we do not use any cost function.

[10]Applying this resampling scheme to the mention pair model did not result in improved performance.

8.3.1.1 Results

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Best first										
Mod. Soon	66.87	**72.38**	69.52	53.86	**63.30**	58.20	53.01	**54.93**	53.95	60.56
Ranking										
Mod. Soon	**72.16**	63.39	67.50	**62.07**	48.36	54.37	51.68	51.21	51.45	57.77
All antec.	70.88	69.10	**69.98**	59.72	58.72	**59.22**	**57.62**	54.16	**55.84**	**61.68**

Table 8.11: Results of a simple mention ranking approach. Highest values for each column are marked bold.

Table 8.11 compares the variants just described with the mention pair model using the modified Soon resampling scheme and best-first clustering. This mention pair model is the variant of the mention pair model from the literature that is conceptually closest to the ranking model.

Adapting the modified Soon scheme from the pair model to ranking leads to a different precision/recall trade-off. While the mention pair model has much higher precision than recall, for the ranking model the reverse holds. Since the drop in precision is more severe than the increase in recall, overall performance is lower.

When allowing the ranking model to learn from all antecedents of non-pruned mentions, precision and recall are much more balanced (with the exception of $CEAF_e$, where recall improves substantially). Overall, this variant of the ranking model performs more than 1 point average F_1 better than the mention pair counterpart, and also almost 1 point average F_1 better than the best-performing mention pair model (see Table 8.7).

8.3.1.2 Error Analysis

The error numbers in Table 8.12 confirm the findings obtained by an analysis of the results of the evaluation metrics. Using the resampling schemes within a ranking model instead of a mention pair model results in improved recall, but decreases precision. Again, as we can see from Figure 8.5, most of the differences can be attributed to differences with regard to the *Unresolved* and *False Anaphoric* errors. There are fewer *Unresolved* recall errors, but more *False Anaphoric* precision errors.

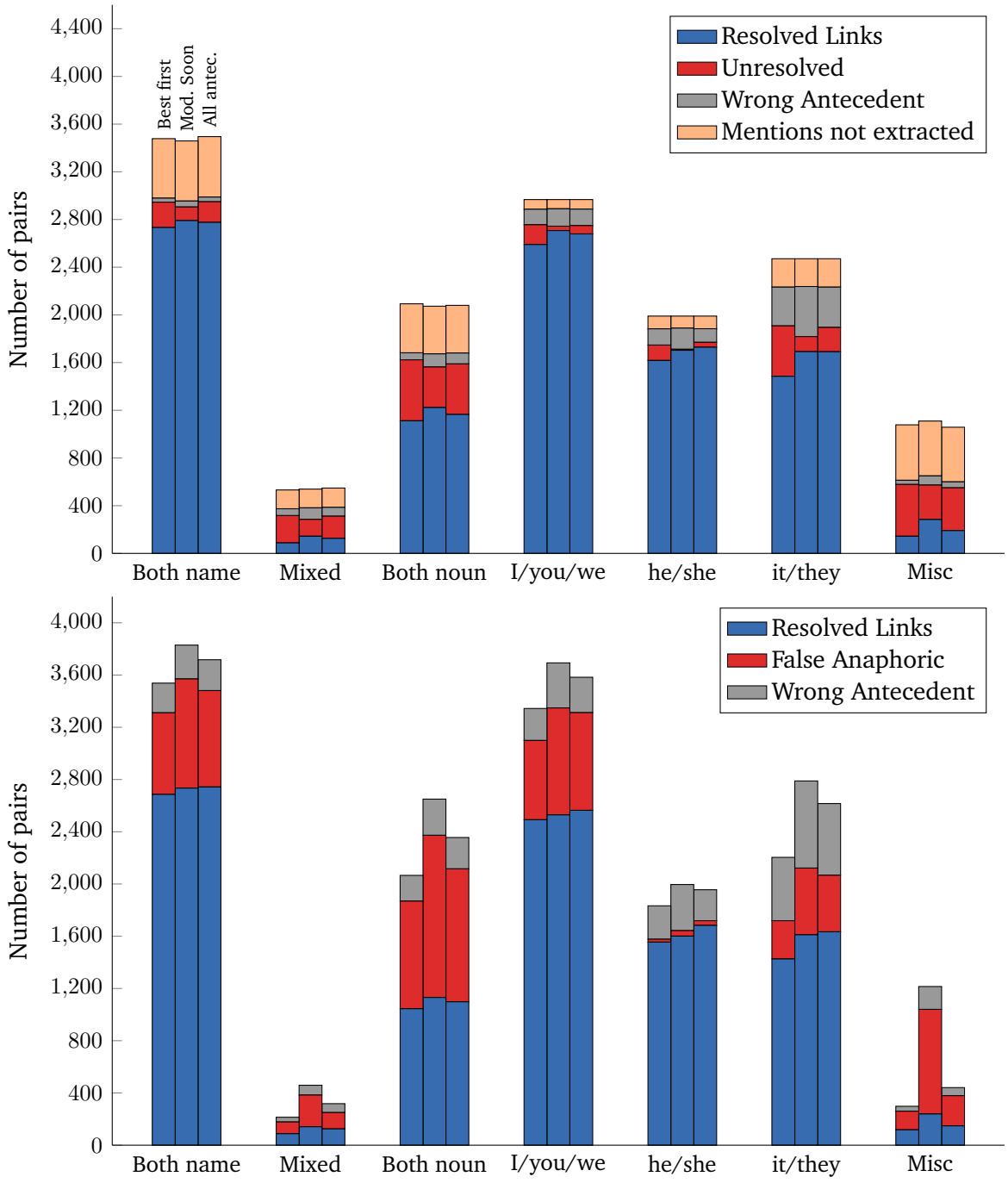

Figure 8.5: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of the simple mention ranking approach on CoNLL-2012 English development data. Left bar: mention pair model with best-first clustering and modified soon resampling; middle bar: simple ranking model with modified Soon resampling; right bar: simple ranking model with all antecedents resampling.

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Best first	4,839	14,609	33%	4,087	13,497	30%
Ranking						
Mod. Soon	4,065	14,609	28%	6,645	16,631	40%
All antec.	4,252	14,609	29%	4,992	14,987	33%

Table 8.12: Overview of recall and precision errors of the simple mention ranking approach.

There are only minor differences with regard to *Wrong Antecedent* errors. They slightly increase when using the ranking approach instead of the mention pair model. This is noteworthy since mention ranking approaches were devised to find better antecedents for anaphoric mentions (Denis and Baldridge, 2008).

Ranking vs. Mention Pair. The error distribution suggests that the ranking model mainly differs from the mention pair model with regard to the handling of anaphoricity. Since the models only differ in the structure they operate on, the difference in the output can be traced back to differences in the structures. Hence, let us go through the structural differences of the models in detail.

When deciding on an antecedent of a mention m_j, the prediction of the ranking model consists of one pair (m_j, m_k). The pair consisting of the mention m_j and its closest correct antecedent serves as the reference pair when updating the parameter vector (see Section 5.5). Hence, the model learns to predict correct antecedents for mentions. The mention pair model, on the other hand, goes through all pairs (m_j, m_k), and predicts a label, either "+" (coreferent) or "−" (not coreferent) for each pair. The reference annotation for updating the parameter vector is the correct label for each pair. Therefore the model learns to predict correct labels for mention pairs. In particular, when the mention pair model erroneously predicts that a pair (m_j, m_k) is coreferent, but instead the pair (m_j, m_i) is coreferent, the model learns that it should predict that (m_j, m_k) is non-coreferent. It only learns to predict that a pair is coreferent when mislabeling the coreferent pair (m_j, m_i). This increases the likelihood that the model predicts that a pair is not coreferent, leading to more *Unresolved* errors. The mention ranking model, however, learns that m_i is a better antecedent for m_j than m_k, and will therefore be less biased towards predicting non-anaphoricity. This is achieved

by the structure the mention ranking model is based on.

All Antecedents vs. Modified Soon Resampling. Compared to the mention pair model, employing the modified Soon resampling scheme for the ranking model leads to a substantial increase of *False Anaphoric* precision errors, particularly for the *Mixed* and *Misc* categories. The increase is much weaker when using the *all antecedents* scheme, which supplies all candidate antecedents for anaphors considered by the modified Soon resampling scheme. Let us compare the approaches in detail.

For the mention ranking model employing the modified Soon resampling scheme, only one correct antecedent (the closest antecedent) is in the set of candidate antecedents. If the model does not predict this antecedent, the prediction is considered as erroneous. Only regarding the closest antecedent as correct can force the model to learn from difficult pairs. Consider, for instance, a coreference chain consisting of the mentions *Moses*, *he* and *Moses*. When employing the modified Soon resampling scheme, the model has to predict that *Moses* has *he* as antecedent. The other antecedent *Moses*, where a string match suffices to establish coreference, is not even among the candidate antecedents. This pushes the model towards making anaphor-antecedent decisions for pairs which are difficult to resolve correctly by the model.

For the *all antecedents* variant, all correct antecedents are among the candidate antecedents. Some of these antecedents may be easier to resolve than the closest antecedent (such as *Moses* in the example above). If the ranking model resolves the anaphor in focus to one of these antecedents, the decision is not incorrect, therefore no update of the parameter vector is conducted. Hence, the model is not forced to assign the closest antecedent as antecedent during training. As we can see from the results and the error analysis, following this strategy increases precision substantially.

8.3.1.3 Discussion

We devised a simple mention ranking approach that conceptually stays close to a well-performing mention pair variant. Adapting the modified Soon resampling scheme results in higher recall and lower precision. We could attribute these differences in results to differences in the structures employed by the approaches. Allowing the ranking model to consider all antecedents of a mention during training results in an improved model, which outperforms all mention pair variants.

8.3.2 Mention Ranking Variants

In the previous section we adapted resampling schemes developed for the mention pair model to mention ranking approaches. However, mention ranking approaches as discussed in Section 6.3 are designed to consider all candidate antecedents of all mentions, therefore not relying on any resampling of the training data.

Hence, we now consider models where we do not apply any pruning to the graphs during training. We consider three variants of such models, which are inspired from the literature.

We first consider a model that does not prune the graphs during training, but still constrains the latent structures consistent with the reference annotation during training. As the models in the last section, it always compares to the *closest* correct antecedent during training. For this variant, we do not use any cost function, since we are interested in the performance of such an architecture when no cost function is employed. Let us dub the model *No cost*. Since it learns from closest antecedents, it is similar to the model proposed by Denis and Baldridge (2008).

The next variant is the same model except that it employs a cost function. After preliminary experiments, we decided on the cost function used by Fernandes et al. (2014) and Björkelund and Kuhn (2014), which has the form

$$c_{\text{rank}}(x, (m_j, m_i), z) = \begin{cases} \lambda_1 & i = 0 \text{ and } m_j \text{ is anaphoric,} \\ \lambda_2 & i > 0 \text{ and the mentions are not coreferent,} \\ 0 & \text{otherwise.} \end{cases} \tag{8.2}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}_{>0}$. We call the model *Cost*.

Finally, we drop any constraints on the antecedents to learn from during training, which leads to the model described by Chang et al. (2012). While the models above always compare the prediction to the pair consisting of the mention in focus and the *closest* correct antecedent during training, the model without constraints compares the prediction to the pair consisting of the mention in focus and the *highest-scoring* correct antecedent according to the current parameter vector. Since such antecedents are called *latent antecedents* in the literature (Chang et al., 2012; Fernandes et al., 2014), we dub the model *Latent*. It employ the same cost function as the *Cost* model.

8.3.2.1 Results

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Resampling	**70.88**	69.10	69.98	**59.72**	58.72	59.22	**57.62**	54.16	55.84	61.68
No cost	63.01	**81.69**	71.15	49.10	**74.09**	59.06	50.46	61.09	55.27	61.83
Cost	69.53	76.05	72.64	58.02	64.00	60.86	54.44	**62.23**	58.08	63.86
Latent	69.47	76.22	**72.69**	58.04	65.10	**61.51**	55.40	61.90	**58.47**	**64.22**

Table 8.13: Results of ranking models with and without cost functions and constraints on structures during training. Highest values for each column are marked bold.

The results of the models are compared in Table 8.13. All mention ranking variants that consider all mentions outperform the ranking model that uses the best-performing resampling scheme described in the previous section. The *No cost* model has very high precision. The precision/recall trade-offs are better for *Cost* and *Latent*. The *Latent* model improves over *Cost* mainly due to higher B^3 precision and higher $CEAF_e$ recall.

8.3.2.2 Error Analysis

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Resampling	4,252	14,609	29%	4,992	14,987	33%
No Cost	5,402	14,609	37%	2,301	11,270	20%
Cost	4,450	14,609	30%	3,528	13,357	26%
Latent	4,459	14,609	31%	3,535	13,315	27%

Table 8.14: Overview of recall and precision errors of ranking models.

Table 8.14 and Figure 8.6 give details on the recall and precision errors of the different models.

No Cost vs. Resampling. The difference between the *No Cost* and *Resampling* model is that the former does not apply any pruning during training, while the latter only

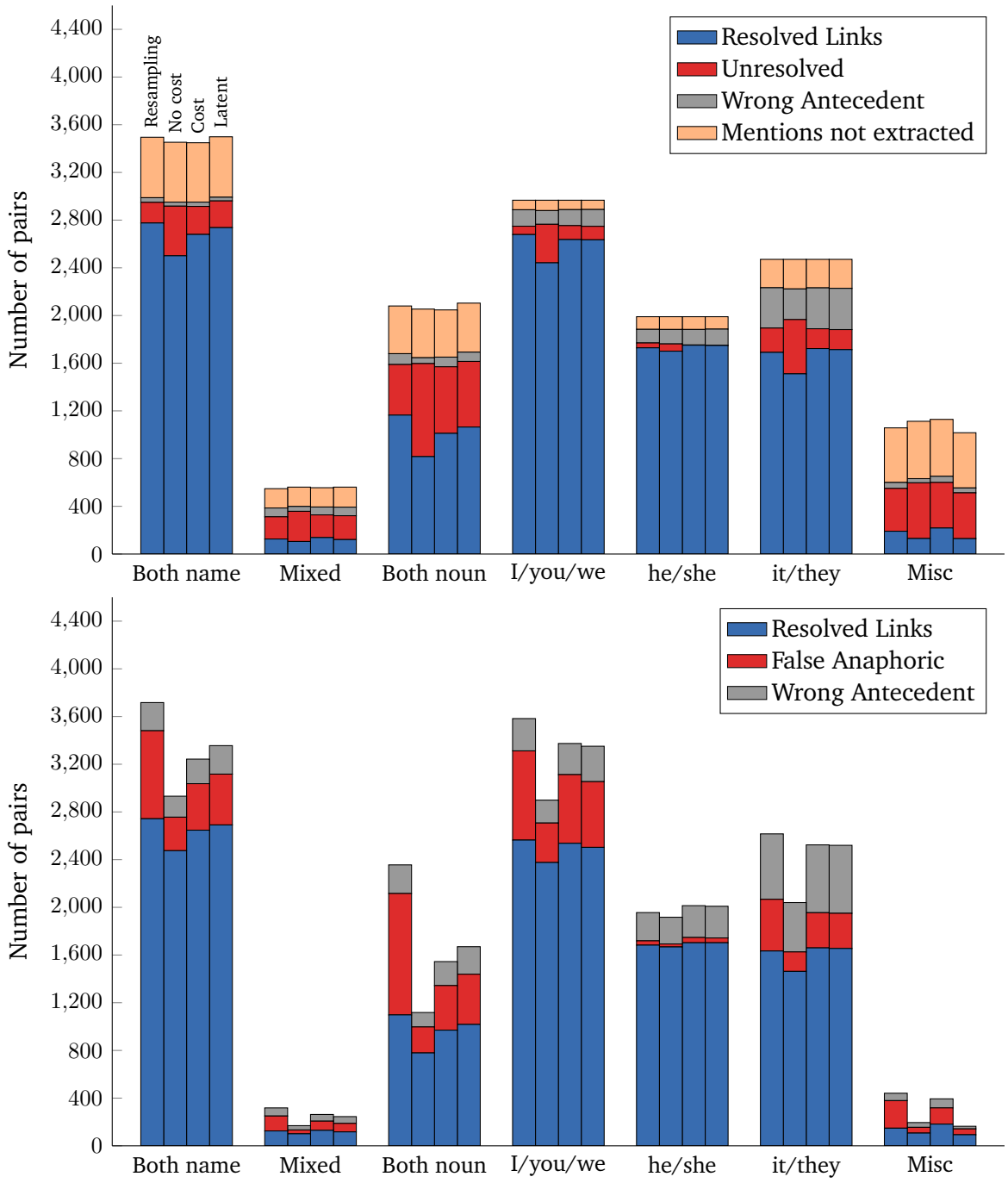

Figure 8.6: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of ranking variants on CoNLL-2012 English development data. First bar: simple ranking model with all antecedents resampling; second bar: ranking model using closest antecedents during training and no cost function; third bar: ranking model using closest antecedents and a cost function; fourth bar: ranking model using latent antecedents and a cost function.

considers mentions as potential anaphors that are in some coreference chain. Hence, the *No Cost* model considers more non-anaphoric mentions. During learning, this guides the model towards proposing that a mention is not coreferent, which leads to more *Unresolved* recall errors but less *False Anaphoric* precision errors. However, the reduction in precision errors is accompanied by a reduction in resolved links.

Cost vs. No Cost. Employing the cost function described by Equation 8.2 results in less recall errors, but more precision errors. For recall, the *Cost* model has a similar error profile as the *Resampling* model. For precision, the model partly retains the *False Anaphoric* error reduction of the *No Cost* model (for example in the *Both noun* category), but has almost as many resolved links as the *Resampling* model. Due to the large number of non-anaphoric mentions, *No Cost* is biased towards determining that mentions are not anaphoric. By using a cost function with appropriate parameters, this bias is mitigated. To summarize, we can attribute the improved performance of the *Cost* model to better anaphoricity detection by using the cost function.

Latent vs. Cost. The *Latent* model differs from the *Cost* model by considering the *highest-scoring* correct antecedent instead of the *closest* correct antecedent when updating during training. For most categories, this change only has a minor effect on performance. The exception is the *Misc* category for precision errors. Most of the errors in this category correspond to links which are considered difficult or unreliable, such as links between a proper name anaphor and a pronoun antecedent (Bengtson and Roth, 2008). By employing highest-scoring instead of closest antecedents during training, the model can avoid learning from these pairs.

8.3.2.3 Discussion

Not pruning the graphs during training improves performance for mention ranking models (we observed the reverse effect for mention pair models because of the imbalanced sample space). Using a suitable cost function improves the precision/recall-trade-off mainly due to improved anaphoricity detection. Employing *latent* instead of closest antecedents during training improves the resolution of pairs deemed difficult.

In our analysis, we could only attribute minor improvements to better antecedent selection for anaphoric mentions. However, since we work on automatically extracted mentions, most mentions we consider are not anaphoric. There may be more sub-

stantial improvements in antecedent selection when considering *gold mentions*, i.e. mentions that are in some coreference chain. We leave an investigation of this setting to future work.

8.3.3 Antecedent Trees

As we saw in Section 6.3, antecedent trees (Fernandes et al., 2014) and mention ranking architectures (Denis and Baldridge, 2008; Chang et al., 2012) are based on the same latent structure. They differ only with respect to the factorization of the latent structure into substructures. While ranking approaches consider the subgraph induced by each anaphor as a substructure, antecedent tree approaches consider the whole document at once.

We obtain an antecedent tree model by dropping the factorization into per-anaphor substructures from the *Latent* mention ranking model discussed above. We dub the model *Tree*.

8.3.3.1 Results

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Latent	**69.47**	76.22	72.69	**58.04**	65.10	**61.51**	**55.40**	61.90	58.47	**64.22**
Tree	68.93	**76.89**	**72.70**	57.35	**65.93**	61.34	55.38	**62.10**	**58.55**	64.20

Table 8.15: Results of an antecedent tree model. Highest values for each column are marked bold.

Table 8.15 compares the results of the *Tree* model with *Latent*, the mention ranking model using latent antecedents. The models differ only with respect to the factorization into substructures. Not factorizing into substructures improves precision and decreases recall. This is consistent with the corresponding experiments for mention pair models (Section 8.2.4).

8.3.3.2 Error Analysis

As shown in Table 8.16 and Figure 8.7, differences in error distributions when changing the substructure factorization are similar to the differences when changing the substructure factorization for mention pair models (Section 8.2.4). However, the differences are less pronounced. For most categories, there are more recall errors and fewer precision errors and correctly resolved links.

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Latent	4,459	14,609	31%	3,535	13,315	27%
Tree	4,537	14,609	31%	3,408	13,098	26%

Table 8.16: Overview of recall and precision errors for antecedent trees.

8.3.3.3 Discussion

Varying the factorization into substructures has only a minor effect on overall performance. Employing a per-document factorization in the mention ranking model, which leads to antecedent trees, improves precision and decreases recall. Analogously to our observations when discussing substructure factorizations for the mention pair model, a per-document factorization ensures slightly more cautious anaphoricity detection.

8.3.4 Graphs

We now consider models that can assign multiple antecedents to mentions. As we have discussed in Section 6.3, such models can be obtained by replacing the *trees* by general *graphs* in the latent structures of the mention ranking and antecedent tree approaches. To our knowledge, such models have not been considered before in the literature.

Hence, we take the *Latent* and *Tree* models described before and replace the latent structure by graphs. The remaining parameters of the models stay the same. Since the models differ only in the factorization into substructures, we name the two models *Graph (Anaphor)* and *Graph (Document)*.

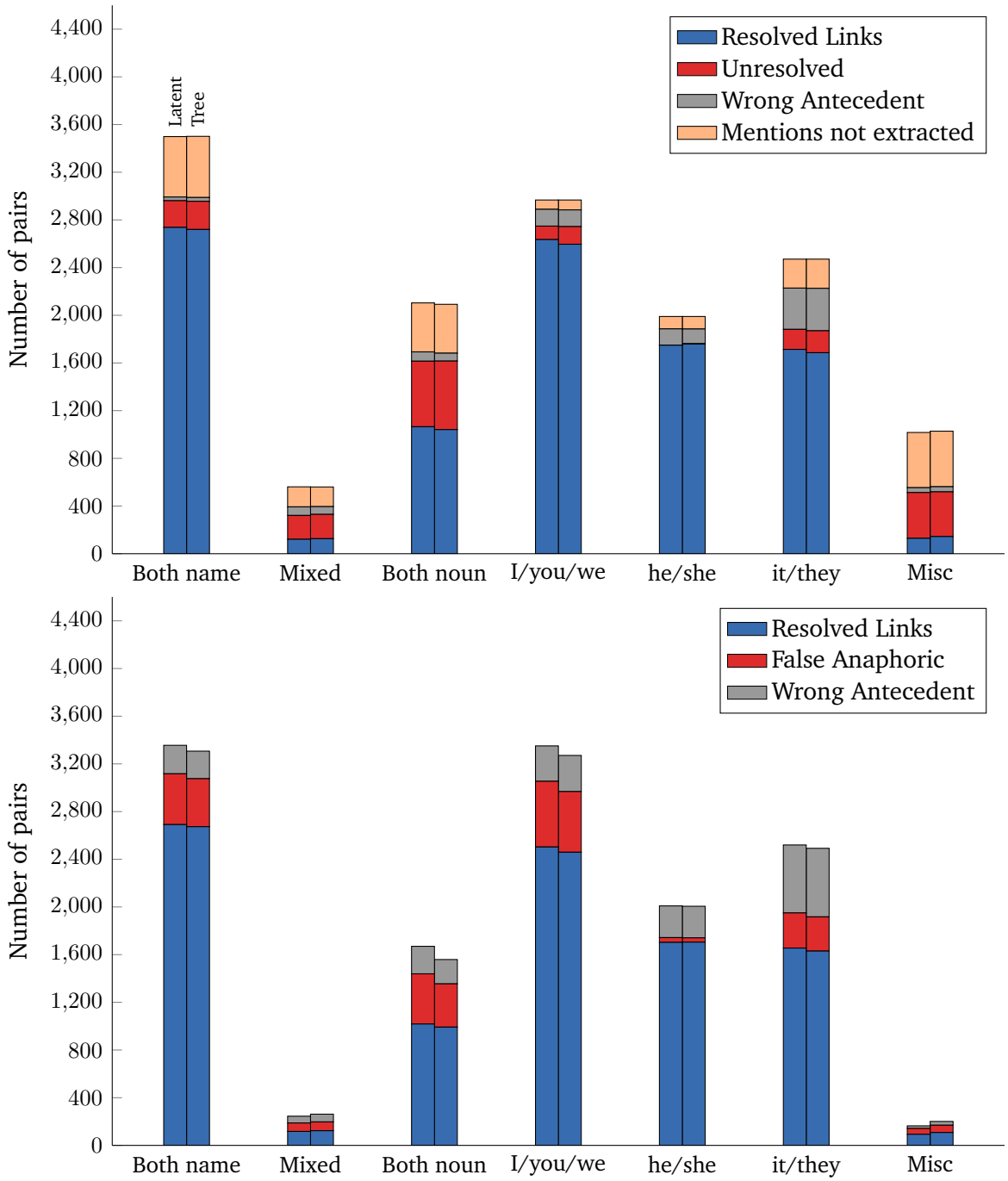

Figure 8.7: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of an antecedent tree model on CoNLL-2012 English development data. Left bar: ranking model with latent antecedents; right bar: antecedent tree model.

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Latent	69.47	76.22	72.69	58.04	65.10	**61.51**	**55.40**	61.90	58.47	**64.22**
Tree	68.93	76.89	**72.70**	57.35	**65.93**	61.34	55.38	**62.10**	**58.55**	64.20
Graph (Anaph.)	**70.26**	73.58	71.88	**60.46**	57.43	58.90	48.99	61.25	54.44	61.74
Graph (Doc.)	55.42	**80.27**	65.57	42.12	65.87	51.39	31.61	61.27	41.70	52.89

Table 8.17: Results of models based on graphs that are not necessarily trees. Highest values for each column are marked bold.

8.3.4.1 Results

Table 8.17 compares the results of the graph-based models to their tree-based counterparts. Switching to graphs as structures increases MUC and B^3 recall for the anaphor-based factorization, and increases MUC precision for the document-wide factorization. Performance according to all other metrics decreases. The drop in performance for the graph-based model with document factorization is especially severe, the performance drops by more than 11 point average F_1 compared to the antecedent tree model.

8.3.4.2 Error Analysis

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Latent	4,459	14,609	31%	3,535	13,315	27%
Tree	4,537	14,609	31%	3,408	13,098	26%
Graph (Anaph.)	4,343	14,609	30%	4,368	13,952	31%
Graph (Doc.)	6,511	14,609	45%	2,487	10,088	25%

Table 8.18: Overview of recall and precision errors for tree- and graph-based models.

Table 8.18 and Figure 8.8 show the errors made by the graph-based models and their tree-based counterparts. For *Graph (Anaphor)*, we can observe a slight decrease in recall errors and a large increase in precision errors. For *Graph (Document)*, recall errors increase substantially while precision errors decrease substantially.

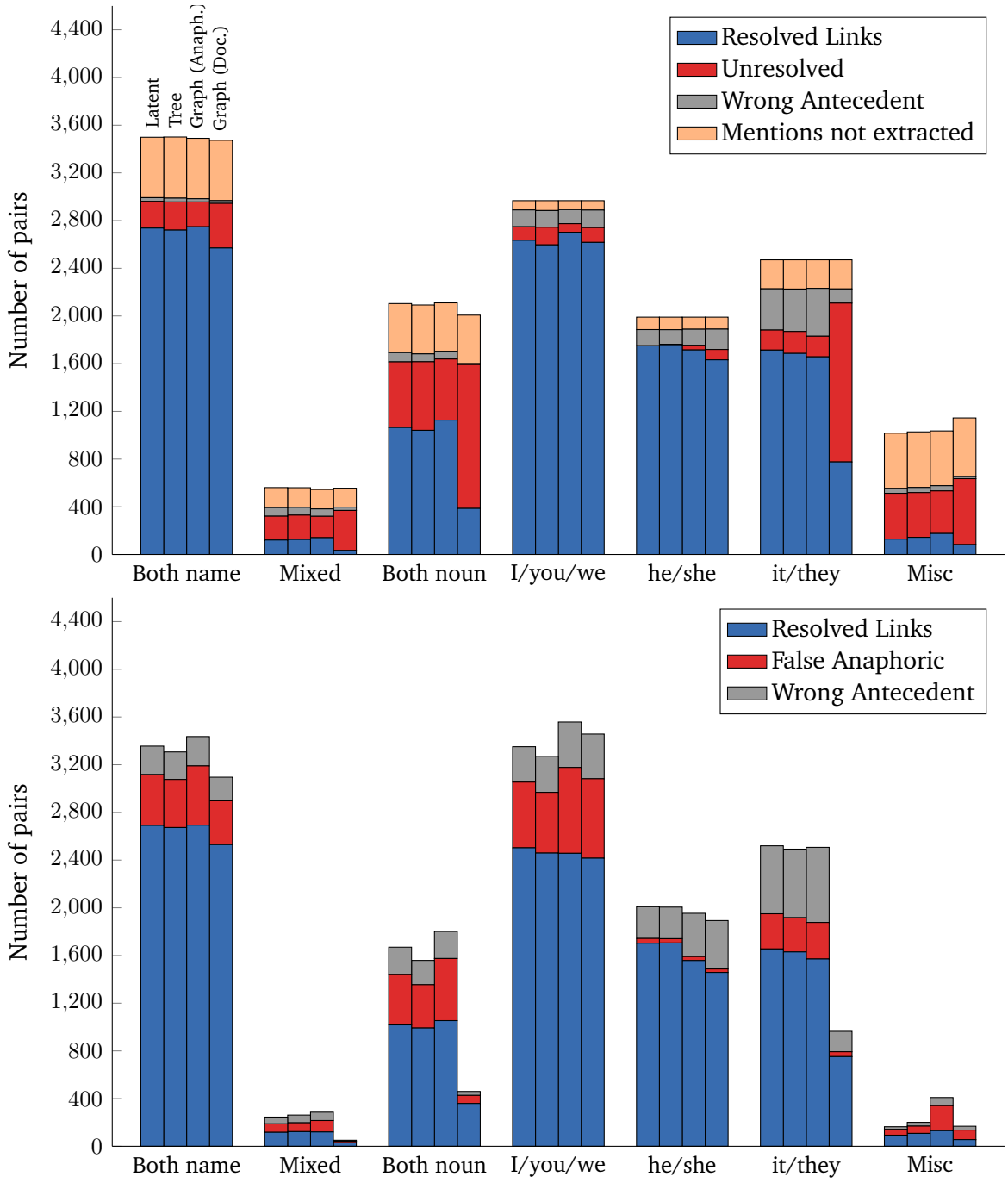

Figure 8.8: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of graph-based models and their tree-based counterparts on CoNLL-2012 English development data. First bar: ranking model with latent antecedents; second bar: antecedent tree model; third bar: graph-based ranking model; fourth bar: document-wide graph-based model.

Graph (Anaphor) vs. Latent. When comparing *Latent* and its graph counterpart *Graph (Anaphor)*, we see that the graph-based model slightly improves recall for *I/you/we*, *Both noun* and *Mixed*. Recall errors for the other categories are unchanged or increased. In particular, we observe more *he/she* errors where the mention is left unresolved. We can observe an increase of precision errors across all categories. Furthermore, for most categories resolved links decrease.

Hence, allowing multiple antecedents for a mention has a slight positive effect on recall, in particular for first- and second-person pronouns. This can be explained by the fact that the data contains broadcast and telephone conversations, in which such pronouns often participate in long coreference chains only consisting of these pronouns. If one mentions gets assigned a wrong antecedent, the chain is broken. Allowing multiple antecedents facilitates avoiding such broken chains. In contrast, recall decreases for categories for which there are strong relationships between anaphors and a single antecedent, such as third-person pronouns. Furthermore, discarding the single-antecedent constraint substantially lowers precision. Since the model has more opportunities for choosing wrong antecedents, this was expected.

Graph (Document) vs. Tree and Graph (Anaphor). For the difference between *Graph (Document)* and *Tree* we can observe a substantial increase in *Unresolved* recall errors, which is accompanied by a reduction in resolved links and precision errors. The only exception is the *I/you/we* category, where there is a slight reduction in recall errors. Compared to *Graph (Anaphor)*, the only difference between the models is the substructure factorization. As we have already seen for the mention pair and tree-based models, a more coarse substructure factorization leads to a more cautious anaphoricity determination. This effect is amplified when allowing multiple antecedents. Except for the recall error reduction for *I/you/we*, this effect also negates the differences we observed when switching from single to multiple antecedents for the model using anaphor-based substructures.

8.3.4.3 Discussion

Graph-based models that drop the single-antecedent constraint improve recall slightly for a few categories, but lead to huge decreases in precision or to a very cautious anaphoricity determination. These observations confirm the reasonability of the single-antecedent constraint.

8.3.5 Summary

Compared to mention pair models, mention ranking models and antecedent trees are based on a structure that directly encodes antecedent decisions. When supplied with a suitable cost function and a suitable search space for antecedents, mention ranking models substantially outperform mention pair models due to better anaphoricity determination. Latent antecedents improve performance for unreliable pairs. Dropping the substructure factorization did not result in better performance due to too cautious predictions. We also investigated the use of general graphs instead of trees, but observed severe performance drops.

8.4 Entity-based Models

The antecedent tree model takes a document-wide perspective on coreference resolution by predicting antecedents for all mentions simultaneously. However, the model does not take any interactions between coreference decisions into account. In order to model such interactions, entity-based approaches (Luo et al., 2004; Yang et al., 2008; Stoyanov and Eisner, 2012; Webster and Curran, 2014; Clark and Manning, 2015, inter alia) perform coreference resolution incrementally and access previous decisions via the structure or via suitable features.

As we have shown in Section 6.4, the gained expressiveness of entity-based models leads to a larger parameter space: they can accommodate more advanced cost functions and features, have variety in the inference schemes and have as additional parameters roll-in and roll-out functions (see Section 5.5). A detailed experimental comparison of these parameters is out of scope for this thesis. We therefore evaluate some selected entity-based models that employ cost functions adapted from the ranking models and only few additional features. This allows us to evaluate and analyze the effect of different ways of adding entity-based information to ranking models. We leave an analysis of the full parameter space of entity-based models to future work.

8.4.1 Assumptions

We first describe assumptions we make for all entity-based models we implement in this thesis.

8.4.1.1 Cost Functions

From the description of structures in Chapter 6 we saw that various cost functions for learning to search for entity-centric models can be devised, ranging from simple edge-factoring cost functions to functions based on coreference resolution evaluation metrics. However, we will only consider cost functions adapted from mention ranking models. This has three reasons: First, the ranking-based cost functions are edge-factored and only depend on the structure of the current time step. This allows us to avoid roll-outs, which greatly increases efficiency (Section 5.5.2.2). Second, cost functions adapted from ranking models have been shown to work well for entity-based models (Björkelund and Kuhn, 2014; Webster and Curran, 2014). Third, while advanced cost functions can be obtained from coreference resolution evaluation metrics, there are several issues: optimizing for the MUC metric can lead to degenerate solutions, and optimizing for the CEAF metrics is not computationally feasible (Stoyanov and Eisner, 2012). A comprehensive treatment of cost functions induced from evaluation metrics requires an analysis of these issues.

Hence, in this thesis, we only employ cost functions adapted from the cost function c_{rank} for ranking models. We discuss limitations of such local cost functions when analyzing the results. We leave an analysis of more advanced cost functions to future work.

Due to the complexity of the entity-based models, training the models is too expensive to optimize hyperparameters by grid search[11]. We therefore do not optimize the cost function hyperparameters for the entity-based models, but instead take the optimal corresponding hyperparameters as determined for the mention ranking model with latent antecedents[12].

8.4.1.2 Transitioning Functions

Other parameters of learning to search approaches are the roll-in transitioning function `in`, which generates structures considered in learning to search step-by-step, and the roll-out transitioning function `out`, which is used for completing deviations of the structures generated by `in`. For the roll-in transitioning function `in`, we follow considerations in previous work (Webster and Curran, 2014) and compare two roll-in

[11] Training the entity-entity models on CoNLL-2012 training data, for example, takes several days.

[12] We ran experiments using the hyperparameters for the antecedent tree model, but this yielded consistently lower results.

transitioning functions:

- $\texttt{in}_{\text{pred}}$ always chooses the highest-scoring latent structure according to the currently learned parameter vector at each step:

$$\texttt{in}_{\text{pred}}(h) = \underset{(h',z') \in \text{Generate}(h)}{\arg\max} \langle \theta, \phi(x, h', z') \rangle. \tag{8.3}$$

  We call $\texttt{in}_{\text{pred}}$ *learned roll-in*, since the latent structures chosen by this transitioning function only depend on the learned parameter vector;

- $\texttt{in}_{\text{ref}}$ always chooses the highest-scoring latent structure that has minimal cost:

$$\texttt{in}_{\text{ref}}(h) = \underset{\substack{(h',z') \in \text{Generate}(h), \\ c(h') = \min_{(h'',z'') \in \text{Generate}(h)} c(h'')}}{\arg\max} \langle \theta, \phi(x, h', z') \rangle. \tag{8.4}$$

  At each step, $\texttt{in}_{\text{ref}}$ makes the optimal decision, in the sense that it is the decision with lowest cost. Therefore, we call $\texttt{in}_{\text{ref}}$ *reference roll-in* or *gold roll-in*.

When using $\texttt{in}_{\text{ref}}$ as the roll-in transitioning function, the learning algorithm learns only from latent structures that have minimal cost. However, this cost information is not available during testing. Therefore, using $\texttt{in}_{\text{pred}}$ exposes the learning algorithm to structures that are more similar to structures that will be encountered during prediction on unseen data.

We only use cost functions with the following property: given any fixed roll-out function, the cost only depends on the structure in the current time step. We therefore do not have to define any roll-out transitioning functions.

8.4.2 Tree-based Mention-Entity Models

In the literature, entity-based models are usually subdivided into *mention-entity* models, which model coreference resolution as attaching mentions to partial entities, and *entity-centric* or *entity-entity* models, which instead model coreference resolution as merging pairs of partial entities.

Since we want to analyze models in order of increasing complexity and expressiveness, we start our analysis of entity-based models with a mention-entity model. In particular, we consider a tree-based mention-entity model, as it relies on the same

structure as mention ranking models and antecedent trees[13]. We saw in Section 6.4.1 that such entity-based models permit various inference schemes. We first consider *left-to-right* inference, as it is computationally the most simple inference scheme. Furthermore, left-to-right inference is the dominant inference scheme for mention-entity models (Luo et al., 2004; Björkelund and Kuhn, 2014; Webster and Curran, 2014). The tree-based mention-entity model with left-to-right inference can be obtained from the general entity-based model relying on trees by using the appropriate `Generate` and `Alternatives` functions as described in Section 6.4.1.

Additionally to the local features described in Section 7.2, the models use the *cluster size* and *antecedent has antecedent* features described in Section 7.3. We use both learned roll-in and gold roll-in. We evaluate the models with the same cost function as for antecedent trees: the cost of a tree is obtained by summing the cost of all edges with respect to c_{rank}. Since the cost function does not depend on any previous or following coreference decision, the cost of a partial latent structure only depends on the current time step.

8.4.2.1 Results

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Tree	**68.93**	76.89	72.70	**57.35**	65.93	61.34	**55.38**	62.10	58.55	64.20
Learned roll-in	68.24	77.25	72.47	56.02	67.45	61.21	55.16	61.45	58.13	63.94
Gold roll-in	68.30	78.22	**72.92**	56.82	67.37	61.64	52.99	**65.80**	**58.70**	**64.42**
+agr.	68.09	**78.28**	72.83	56.69	**67.51**	61.63	52.81	65.70	58.55	64.34
No strct. feat.	68.28	78.21	72.91	56.90	67.32	**61.67**	52.95	65.59	58.59	64.39

Table 8.19: Results of variations of tree-based mention-entity models. Highest values for each column are marked bold.

Table 8.19 compares variants of tree-based mention-entity models. We see that extending antecedent trees with entity-based features does not result in improved performance when using learned roll-in. When using gold roll-in, the entity-based information helps, performance increases by roughly 0.2 points average F_1, mainly due to improved precision.

[13]Due to their low performance, we do not further consider models that are based on general graphs.

We also test the effect of features. The tree-based mention-entity model uses all mention and pairwise features also used by the non-entity-based models, and additionally uses features that describe properties of partial entities. However, the features for partial entities are not induced from mention or pairwise features. In preliminary experiments, we confirmed results that features induced from local features do not improve performance when used on top of the local feature set (Björkelund and Kuhn, 2014; Clark and Manning, 2015). In the line marked "+agr.", we exemplarily evaluate the contribution of the *agreement* feature. As we can see, using this feature on top of the model does not lead to improved performance, the pairwise links already seem to capture the necessary information. Lastly, we consider the impact of the structural features. The line marked "No strct. feat." shows the performance of the model when the structural feature *antecedent has antecedent* is removed. As we can see, the structural feature only has an insignificant impact on performance.

8.4.2.2 Error Analysis

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Tree	4,537	14,609	31%	3,408	13,098	26%
Learned roll-in	4,638	14,609	32%	3,284	12,907	25%
Gold roll-in	4,630	14,609	32%	3,124	12,756	24%

Table 8.20: Overview of recall and precision errors for tree-based mention-entity models.

In Table 8.20 and Figure 8.9 we show the errors of tree-based mention-entity models with learned and gold roll-in. We compare the errors with errors of an antecedent tree model which uses the same mention and pairwise features. We do not include the feature variants of the model with gold roll-in in the comparison, since the difference in performance was marginal.

From the figure we can see that the most obvious difference is a reduction in precision errors, in particular *False Anaphoric* errors, for the tree-based mention entity model with gold roll-in. This reduction is accompanied by a slight increase in *Unresolved* recall errors.

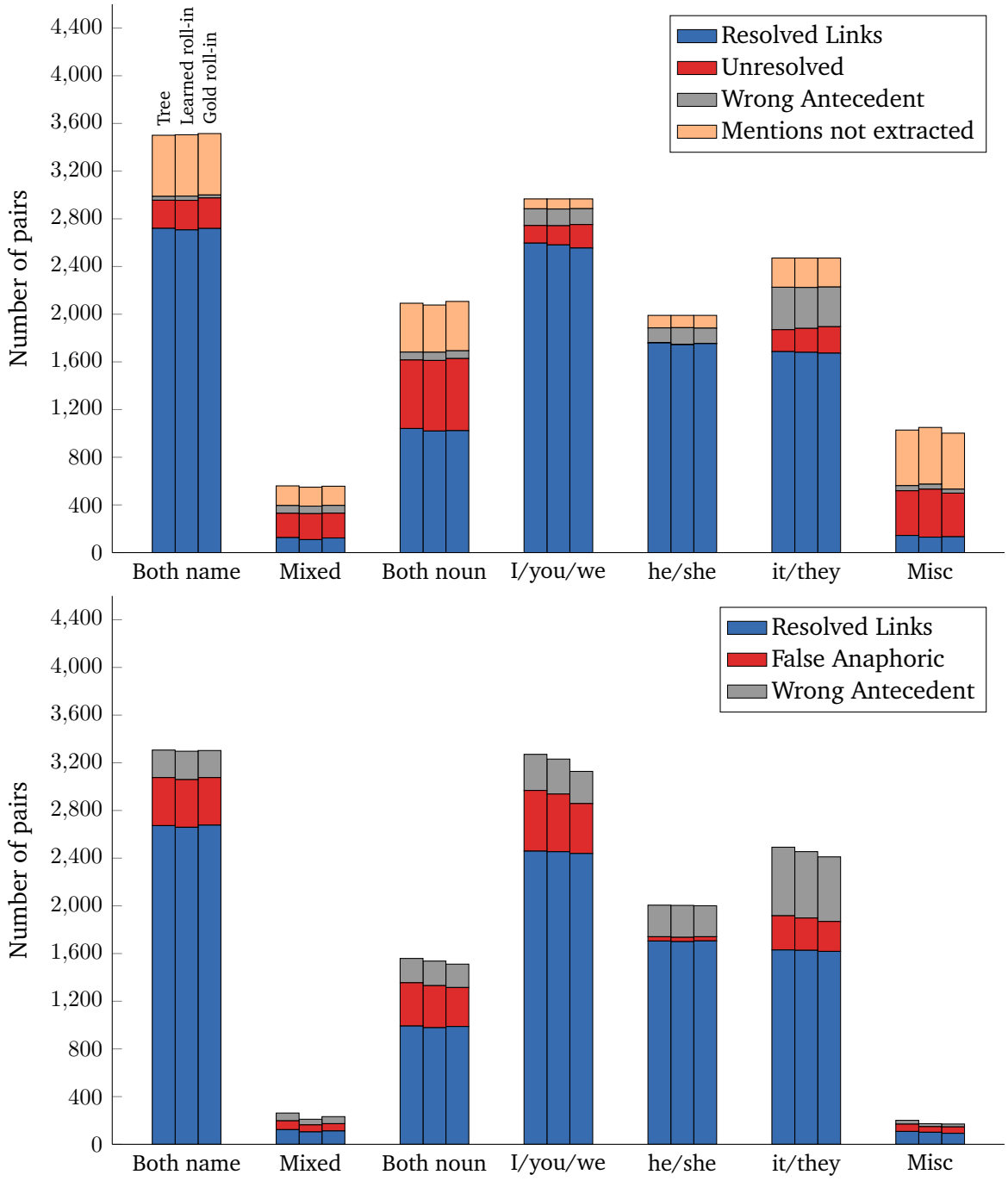

Figure 8.9: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of tree-based mention-entity models on CoNLL-2012 English development data. Left bar: antecedent tree model; middle bar: tree-based mention-entity model with learned roll-in; right bar: tree-based mention-entity model with gold roll-in.

This suggests that the entity-level features we employ in the model – the size of the cluster and whether the candidate antecedent has an antecedent – are helpful for determining anaphoricity. However, there is no significant use of this features for *Wrong Antecedent* errors. The model that employs learned roll-in shows similar trends, but the differences are less pronounced.

8.4.2.3 Discussion

Tree-based mention-entity model are able to augment the modeling of anaphor-antecedent relations with entity-based information. Adding features that describe properties and structure of partial entities improves performance with respect to anaphoricity when using gold roll-ins. These results suggest that we should further explore entity-based features that aim to improve choosing the correct antecedent.

Following much previous work (Webster and Curran, 2014; Björkelund and Kuhn, 2014), the cost function we used for tree-based mention-entity models evaluates only individual edges. We suspect that this property is responsible for the low performance of the model when using learned roll-ins. When using learned roll-ins, it may happen during training that none of the preceding partial entities only contains coreferent mentions. However, the local cost function cannot distinguish between different partial entities, since it only considers anaphor-antecedent pairs. More advanced cost functions, such as measures induced from coreference resolution evaluation metrics (as used by Stoyanov and Eisner (2012) and Clark and Manning (2015)) do not have this problem and may improve performance.

8.4.3 Tree-based Mention-Entity Easy-first Models

Left-to-right inference is the simplest and most restrictive inference scheme for entity-based models. We now study the effect of switching to *easy-first* inference. During easy-first inference, we do not process the first unattached mention, but allow to consider any unattached mention. In our framework, this is modeled by dropping the corresponding constraints for the `Generate` and `Alternatives` functions. To the best of our knowledge, mention-entity models with easy-first inference were not studied before.

As we have described in Section 6.4.1, easy-first mention-entity models have a very large search space that must be restricted in order to perform efficient learning and inference. We restrict the search space by applying a simple heuristic. Note that, given

a parameter vector θ and a document x, we can compute the *local* features that only describe a pair of mentions and the weights of these features a priori. In contrast, we cannot compute entity-based features a priori, since these depend on the coreference decisions for x. We therefore use the scores of the pairs to restrict the search space: given a mention m_j, consider only the k highest-scoring antecedents m_i according to the scoring function $(m_j, m_i) \mapsto \langle \theta, \phi(x, (m_j, m_i), z) \rangle$. We experimented with $k = 1$, 5, 10, 15, ... and found that setting k larger as 20 led to prohibitively long running times. We therefore set k to 20.

8.4.3.1 Results

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Left-to-right										
Lrnd. roll-in	68.24	77.25	72.47	56.02	**67.45**	61.21	55.16	61.45	58.13	63.94
Gold roll-in	68.30	**78.22**	**72.92**	56.82	67.37	**61.64**	52.99	**65.80**	58.70	**64.42**
Easy-first										
Lrnd. roll-in	68.24	77.07	72.39	55.93	67.25	61.07	**55.22**	61.80	58.33	63.93
Gold roll-in	**68.71**	77.41	72.80	**57.01**	66.54	61.41	54.30	64.06	**58.78**	64.33

Table 8.21: Results of variations of tree-based easy-first mention-entity models. Highest values for each column are marked bold.

In Table 8.21 we compare tree-based easy-first inference models with their left-to-right inference counterparts. Compared to the corresponding left-to-right model, easy-first inference performs slightly worse with both learned and gold roll-ins, mainly due to lower precision.

8.4.3.2 Error Analysis

Table 8.22 and Figure 8.10 compare errors for tree-based mention-entity models with left-to-right and easy-first inference. There are only small differences between the models when employing the same roll-in function. Compared to the other variants, the mention-entity model with left-to-right inference and gold roll-ins makes fewer precision errors for *I/you/we* and *it/they*. The easy-first model with gold roll-in shows slightly improved recall for the same categories. The models with easy-first inference

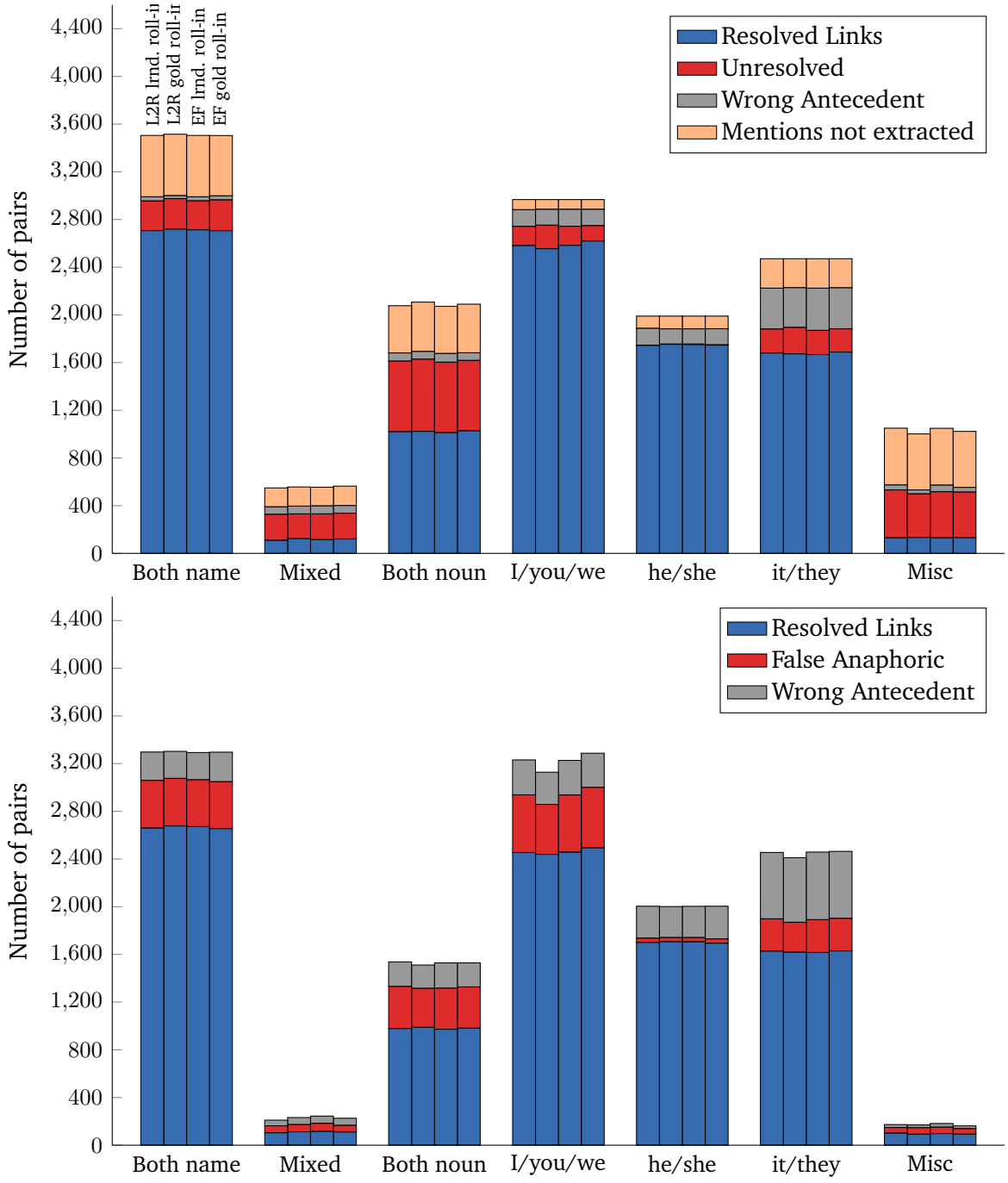

Figure 8.10: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of tree-based mention-entity models with different inference schemes on CoNLL-2012 English development data. First bar: left-to-right inference with learned roll-in; second bar: left-to-right inference with gold roll-in; third bar: easy-first inference with learned roll-in; fourth bar: easy-first inference with gold roll-in.

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Left-to-right						
Learned roll-in	4,638	14,609	32%	3,284	12,907	25%
Gold roll-in	4,630	14,609	32%	3,124	12,756	24%
Easy-first						
Learned roll-in	4,638	14,609	32%	3,303	12,936	26%
Gold roll-in	4,570	14,609	31%	3,315	12,968	26%

Table 8.22: Overview of recall and precision errors for tree-based mention-entity models with left-to-right and easy-first inference.

are devised to make more reliable decisions first. The error profile does not indicate that the models achieve this.

8.4.3.3 Discussion

With the exception of better recall for first- and second-person pronouns and third-person ungendered pronouns, we could not observe improved performance when using easy-first inference. However, our models are constrained by heuristics for restricting the search space and by the cost function employed. As the cost function assigns costs based on individual anaphor-antecedent decisions, it only has a very local notion of "easiness" or "reliability" of a coreference decision. For the reasons mentioned in Section 8.4.1, we leave a detailed investigation of global cost functions to future work.

8.4.4 Tree-based Entity-Entity Models

We now consider entity-entity models that are based on a tree structure. While mention-entity models approach coreference resolution by attaching mentions to partial entities, entity-entity models (Culotta et al., 2007; Stoyanov and Eisner, 2012; Clark and Manning, 2015) instead consider *pairs* of partial entities at each step. We can obtain tree-based entity-entity models from tree-based easy-first mention-entity models by extending the scope of the features to pairs of partial entities instead of pairs of a mention and a partial entity. The remaining parameters such as the cost function or the heuristic to restrict the search space remain the same.

8.4.4.1 Results

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Easy-first										
Lrnd. roll-in	68.24	77.07	72.39	55.93	67.25	61.07	**55.22**	61.80	58.33	63.93
Gold roll-in	68.71	**77.41**	72.80	57.01	66.54	61.41	54.30	64.06	**58.78**	**64.33**
Entity-entity										
Lrnd. roll-in	68.44	77.35	72.62	56.62	**67.87**	**61.74**	55.01	62.37	58.46	64.27
Gold roll-in	**69.14**	77.23	**72.96**	**57.23**	66.06	61.32	53.03	**65.12**	58.45	64.24

Table 8.23: Results of entity-entity models. Highest values for each column are marked bold.

In Table 8.23 we compare the tree-based entity-entity model with its mention-entity variant. Using the entity-entity model when learning with learned roll-in performs roughly 0.3 average F_1 better than the mention-entity model. The entity-entity model with gold roll-in performs slightly worse than its mention-entity counterpart.

8.4.4.2 Error Analysis

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Mention-entity						
Learned roll-in	4,638	14,609	32%	3,303	12,936	26%
Gold roll-in	4,570	14,609	31%	3,315	12,968	26%
Entity-entity						
Learned roll-in	4,609	14,609	32%	3,278	12,927	25%
Gold roll-in	4,507	14,609	31%	3,393	13,079	26%

Table 8.24: Overview of recall and precision errors for entity-entity models.

In Table 8.24 and Figure 8.11, the errors of the model variants are summarized and compared to errors of the tree-based mention-entity model with easy-first inference. We can only observe minor differences between the error profiles when considering the same roll-in function.

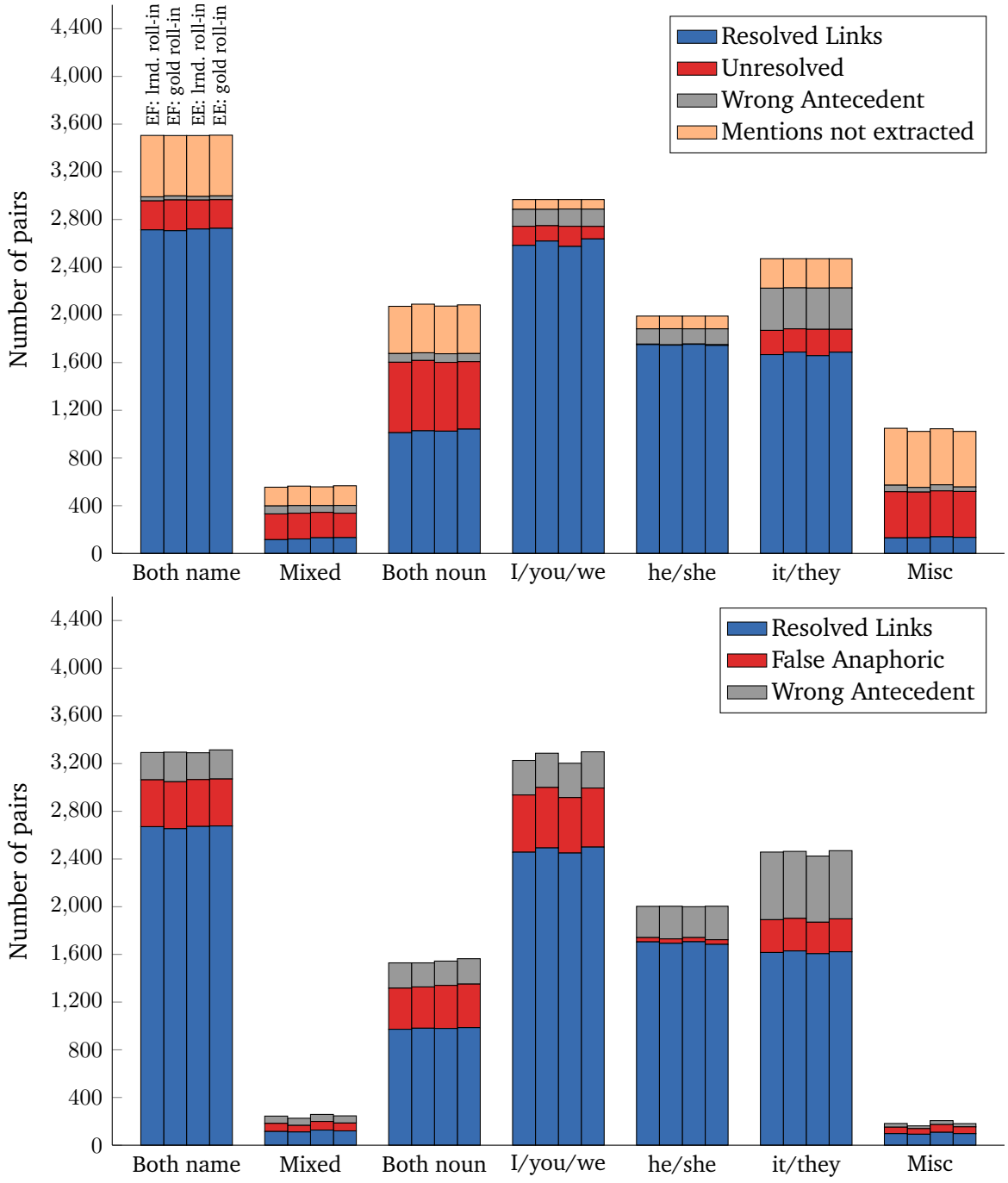

Figure 8.11: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of tree-based entity-entity models on CoNLL-2012 English development data. First bar: mention-entity model with easy-first inference and learned roll-in; second bar: mention-entity model with easy-first inference and gold roll-in; third bar: entity-entity model with learned roll-in; fourth bar: entity-entity model with gold roll-in.

8.4.4.3 Discussion

The entity-entity model performs similarly to the mention-entity model with easy-first inference. With gold roll-in the performance is slightly worse, while there is an improvement when using learned roll-ins. This indicates that the entity-entity model may be able to leverage information about predicted coreference chains, but we could not observe any substantial differences in the error profiles.

The performance of the entity-entity models is still below the performance of the left-to-right mention-entity model with gold roll-in and is only slightly better than the performance of the latent ranking model. As was the case with mention-entity models, this may partly be caused by the cost function, which only evaluates the local quality of a coreference decision[14]. Again, for the reasons mentioned in Section 8.4.1, we leave a detailed investigation of global cost functions to future work.

8.4.5 Hypergraph Mention-Entity Models

We finally consider entity-based models that are based on a *hypergraph* representation (Section 6.4.2). These models are unable to access information about the internal structure of partial entities, such as anaphor-antecedent decisions. As mention-entity models with left-to-right inference performed best for the tree structure, we only consider the corresponding model for hypergraphs. The model can be obtained by considering appropriate `Generate` and `Alternatives` functions, as described in Section 6.4.2.

Hypergraph-based mention-entity models cannot use the local features described in Section 7.2 directly, since these rely on mention pairs (m, n), whereas hypergraph-based mention-entity models score hyperedges, which are pairs consisting of a mention and a partial entity. As described in Section 7.3.1, we induce hyperedge features from local features by either evaluating the closest mention or by applying logical predicates. Besides the adapted local features, we also use the *cluster size* feature described in Section 7.3.2. We cannot use the structural features (Section 7.3.3), since hypergraph-based mention-entity models do not model structure in the partial entities.

Hypergraph-based models admit variation regarding the cost function. Since we only consider edge-factoring cost functions, we evaluate three variants, where two are

[14] Clark and Manning (2015, 2016) report improvements of an entity-entity architecture over a ranking architecture. However, they use different features, cost functions and heuristics for restraining the search space. In this thesis, we deliberately kept these parameters very simple.

induced from mention-ranking cost functions as described in Section 6.4.2:

- the first variant, *No cost*, does not utilize any cost function;
- the second variant, *Hyper cost*, computes the cost of an hyperedge $(\{m_j\}, X)$ by aggregating the ranking cost function via

$$c_{\text{aggr}}(x, (\{m_j\}, X), z) = \sum_{m_i \in X} c_{\text{rank}}(x, (m_j, m_i), z); \tag{8.5}$$

- the third variant, *Pair cost*, computes the cost of an hyperedge $(\{m_j\}, X)$ by applying the ranking cost function c_{rank} described by Equation 8.2 as follows:

$$c_{\text{max}}(x, (\{m_j\}, X), z) = \max_{m_i \in X} c_{\text{rank}}(x, (m_j, m_i), z). \tag{8.6}$$

To ensure that the cost function only depends on the latent structure at the current time step, we set $c_{\text{aggr}} \equiv c_{\text{max}} \equiv 0$ for all edges except the one added in the current time step.

Finally, recall that we extended mention features and distance features to entity-level features by applying them to the pair (m_j, m_i) consisting of the mention in focus and the *closest* mention in the preceding partial entity (Section 7.3)[15]. When training with learned roll-ins, we can not ensure that (m_j, m_i) consists of coreferent mentions for the minimum-cost highest-scoring hyperedge. Hence, in order to learn reasonable weights for the mention features and distance features, we train the hypergraph-based models only with gold roll-ins.

8.4.5.1 Results

Table 8.25 shows the results of the hypergraph models with different cost functions. The numbers are compared with the results of the tree-based mention-entity model using left-to-right inference. The *No cost* variant has highest precision, *Pair cost* improves mainly in recall. *Hyper cost* has the lowest performance. The best hypergraph model, *Pair cost*, performs better than the mention pair models, but is roughly 1 point average F_1 worse than the best ranking-based approach.

[15] Other variants of applying the features resulted in lower performance.

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Left-to-right (Tree)	**68.30**	78.22	**72.92**	**56.82**	67.37	**61.64**	52.99	65.80	**58.70**	**64.42**
No cost	62.36	**81.40**	70.62	49.36	**71.83**	58.51	46.79	**66.78**	55.02	61.38
Hyper cost	63.24	77.28	69.56	47.57	70.97	56.96	**53.76**	58.92	56.22	60.91
Pair cost	66.84	78.17	72.06	54.73	67.25	60.35	50.74	65.81	57.30	63.24

Table 8.25: Results of hypergraph-based mention-entity models with and without cost functions. All models use gold roll-in. Highest values for each column are marked bold.

8.4.5.2 Error Analysis

	Recall			Precision		
Model	**Errors**	**Max**	**% of Max**	**Errors**	**Max**	**% of Max**
Left-to-right	4,630	14,609	32%	3,124	12,756	24%
No cost	5,497	14,609	38%	2,523	11,193	23%
Hyper cost	5,369	14,609	37%	3,030	11,956	25%
Pair cost	4,843	14,609	33%	3,258	12,492	26%

Table 8.26: Overview of recall and precision errors for hypergraph-based mention-entity models.

The errors of the different configurations of the hypergraph-based models are summarized in Table 8.26 and Figure 8.12. Recall and precision errors are extracted by spanning tree algorithms based on accessibility, since the hypergraph approaches neither output pairwise scores nor provide output spanning trees. Hence, when comparing errors for individual categories between hypergraph-based models and the tree-based model, we have to keep in mind that we use different notions of an error. Nevertheless, we can still compare total error numbers and we can get an assessment of the distribution of errors.

Let us first consider the numbers displayed in Table 8.26 in detail. Analogously to the mention ranking model, the *No cost* variant makes fewer precision errors than the other variants, at the expense of many recall errors. The cost function employed

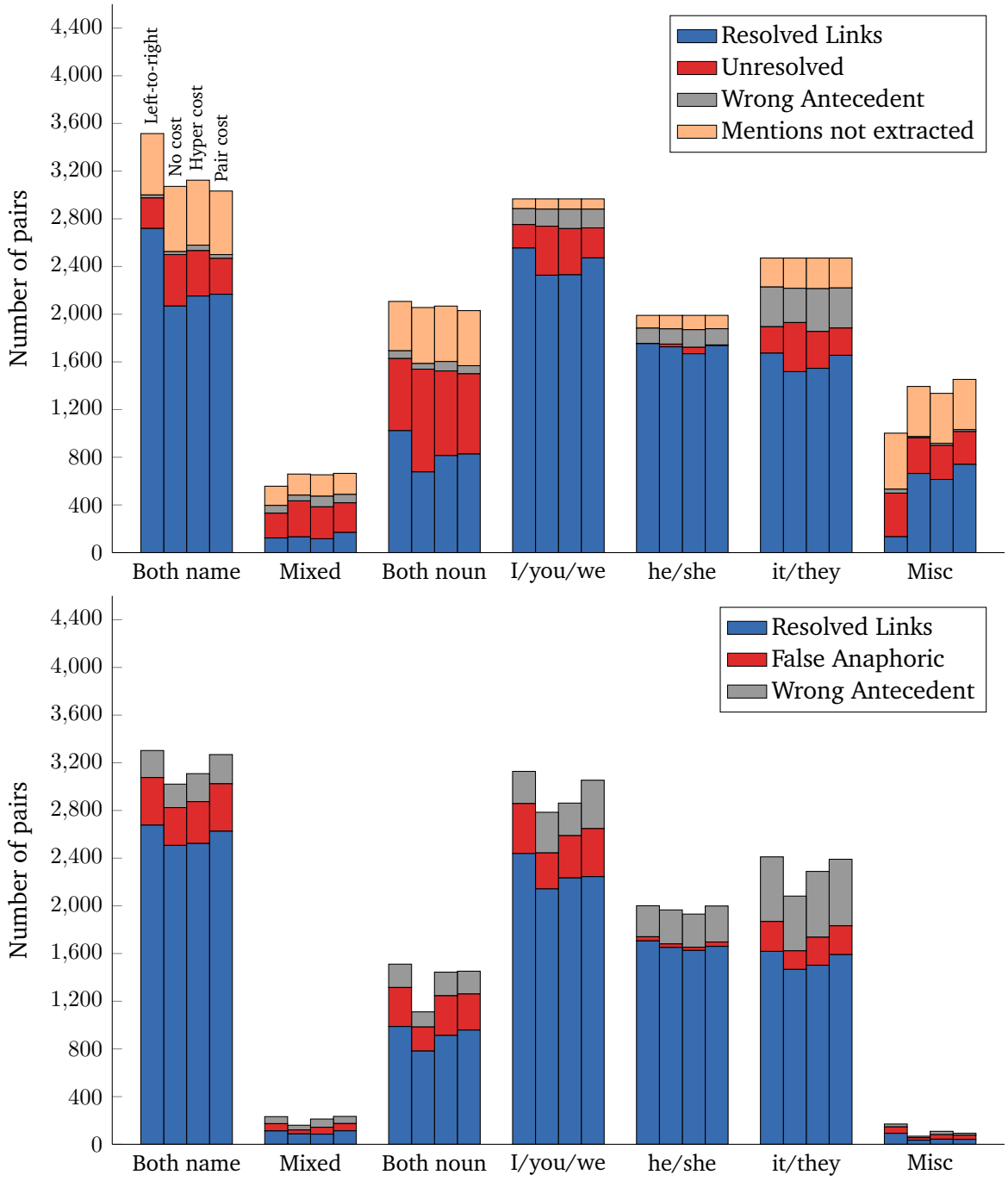

Figure 8.12: Recall errors and resolved links (top) and precision errors and resolved links (bottom) of hypergraph-based mention-entity models on CoNLL-2012 English development data. First bar: antecedent tree model; second bar: hypergraph model without a cost function; third bar: hypergraph model with *Hyper cost*; fourth bar: hypergraph model with *Pair cost*.

by the *Hyper cost* model, which is sensitive to the cluster size, also leads to cautious predictions. For *Pair cost*, the size of the partial entity does not matter, and the number of predictions is more similar to the tree-based model.

We now compare the errors for individual categories as displayed in Figure 8.12. Compared to the mention-entity model based on trees, *Unresolved* recall errors are increased for all categories except for *Misc*. For precision, the hypergraph models have less resolved links, slightly less *False Anaphoric* errors and, for most categories, more *Wrong Antecedent* errors. For both recall and precision errors, the effects are least pronounced for *Pair cost*. The error profile of this variant is similar to the mention-entity model based on trees.

For the *No cost* and *Hyper cost* variants, the increase in *Unresolved* and the decrease in *False Anaphoric* errors can be explained by the cautious predictions taken by the models due to the cost function used (or the absence of a cost function). The model using *Pair cost* makes a similar number of predictions as the antecedent tree model. However, the number of *Wrong Antecedent* errors is increased (as for the other hypergraph variants). In contrast to the models considered so far in this chapter, the hypergraph-based models do not have access to individual anaphor-antecedent relations. Following the literature (e.g. Rahman and Ng, 2011a), the majority of features for the hypergraph models where induced by features devised for mention pairs. Our analysis suggests that using such features leads to a decrease in performance, in particular due to worse antecedent selection.

8.4.5.3 Discussion

Hypergraph-based mention-entity models can aggregate information about the relation of a mention and all mentions in a preceding partial entity. However, fundamentally different from the models we considered so far in this thesis, hypergraph-based models cannot access information about individual anaphor-antecedent decisions. At least with the standard feature set considered in the literature, not being able to harness this information leads to a decrease in performance.

8.4.6 Summary

We considered two variants of entity-based models: mention-entity models and entity-entity models. Mention-entity models differ from ranking and antecedent tree models

by incorporating entity-based information when attaching a mention. In our evaluation and analysis, we only considered few features and local cost functions. A left-to-right tree-based mention-entity model improved slightly when using entity-based information. Learning from noisy partial entities, as obtained by learned roll-in, employing easy-first inference or leveraging information about pairs of entities did not improve performance further. We suspect that this is caused mainly by the narrow scope of the cost functions we employ. We found that hypergraph-based models based on a standard feature set cannot improve performance, which we attribute to the difficulty of predicting coreference relations by aggregating mention and pairwise features.

8.5 Evaluation on Test Data

We now conclude our experiments by evaluating the models on test data. To do so, we train the models with optimal hyperparameters (as determined on development data) on the concatenation of training and development data. We compute statistical significance of differences in MUC, B^3 and $CEAF_e$ F_1 score using an approximate randomization test (Noreen, 1989). We say that a difference is *statistically significant* if the p-value of the corresponding test is below 0.05.

	MUC			**B^3**			**$CEAF_e$**			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
nn_coref	69.31	76.23	72.60	55.83	66.07	60.52	54.88	59.41	57.05	63.39
Stanford Sieve	64.26	65.19	64.72	49.09	56.84	52.68	52.54	46.55	49.73	55.59

Table 8.27: Results of *nn_coref* (Wiseman et al., 2015) and *Stanford Sieve* (Lee et al., 2013) on CoNLL-2012 English test data.

Table 8.27 shows the performance of the state-of-the-art system *nn_coref* (Wiseman et al., 2015) and of the widely used *Stanford Sieve* (Lee et al., 2013) on CoNLL-2012 test data. Compared to the results on development data (Table 8.2), the overall performance drops by roughly 1 point average F_1.

8.5.1 Results

Table 8.28 shows the results of selected models on CoNLL-2012 test data[16]. Similar to StanfordSieve and nn_coref, most models drop by roughly 1 point F_1 score. The results confirm the trends we observed on development data. In particular, the mention ranking models with latent antecedents performs very well. It is only outperformed by the tree-based mention-entity model with left-to-right inference.

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Av. F_1**
Vanilla pair										
All pairs	66.43	77.06	71.35	54.77	51.52	53.09	34.48	61.22	44.12	56.19
Best first										
All pairs	62.81	78.65	69.84*†	47.32	64.56	54.61*†	41.11	59.41	48.59*†	57.68
Mod. Soon	67.16	71.63	69.32†	52.30	60.40	56.06*†	50.61	51.20	50.90*†	58.76
Ranking										
All antec.	**71.61**	68.55	70.05*	**58.60**	56.74	57.65*	**56.23**	51.70	53.87*	60.52
No cost	62.91	**81.23**	70.91	46.89	**72.96**	57.09	48.67	58.99	53.34	60.45
Cost	69.62	76.26	72.79*†	56.10	63.43	59.54*†	51.97	59.61	55.53*†	62.62
Latent	69.48	76.47	72.81†	55.81	65.25	60.16*†	53.39	60.16	56.58*†	63.18
Antec. Tree	68.63	77.27	72.69†	54.83	66.38	60.05†	52.63	60.05	56.10*†	62.95
Mention-Ent.										
Tree[17]	68.28	78.42	**73.00**	55.14	66.73	**60.38**	50.47	**64.54**	**56.64***†	**63.34**

Table 8.28: Results of models on CoNLL-2012 English test data. Highest values for each column are marked bold. * indicates significants differences in F_1 score compared to the preceding model in the table; † indicates significant differences compared to each model's *baseline*, which is defined as *Vanilla pair: All pairs* for the mention pair models, *Ranking: No cost* for the ranking/tree models, and *Antec. Tree* for the entity-based model.

Regarding the statistical significance of the differences in F_1 scores, we compare each model with the model that is immediately preceding in the table, since the models are in order of increasing complexity. Furthermore, we defined for each model a baseline to which we compare additionally.

We find that switching from aggressive-merge clustering to best-first clustering yields

[16]Results for all models on test data can be found in Appendix A.
[17]Using gold roll-in and left-to-right inference.

significantly improved performance according to all metrics. Employing the *Modified Soon* scheme instead of learning all pairs improves performance significantly for all metrics expect for MUC. The switch from the mention pair model to ranking again yields significantly improved performance according to all metrics. Learning from all mentions in the ranking model (as in the *No cost* model) leads to no significants improvements over learning from instances obtained by the *All antecedents* resampling scheme. Using a cost function and employing latent antecedents improves performance significantly according to most metrics. With the exception of the CEAF_e metric, there are no significant differences when using antecedent trees instead of mention ranking or when extending the tree-based model with entity-based information.

8.5.2 Discussion

On development data, we explained differences in the output of models using a detailed error analysis, and by relating the errors to the modeling assumptions of the individual models and to the differences in structures and remaining parameters between the models. The evaluation metric results on test data confirm the trends we observed on development data. Significant effects on differences in F_1 score were obtained by varying parameters of the mention pair models, switching from the mention pair model to the ranking model, by introducing a cost function to the ranking model, and by switching from closest antecedents during learning to latent antecedents. The remaining changes affected the results, but the effects we analyzed in this chapter seem to be too small to lead to a statistically significant difference according to most metrics.

8.5.3 Comparison with the State-of-the-Art

In Table 8.29, we compare the best-performing models with the state-of-the-art system *nn_coref* of Wiseman et al. (2015). The models perform similarly to *nn_coref*, the difference in results is not statistically significant.

8.6 Summary

In this chapter, we evaluated many structures for coreference resolution, ranging from simple mention pair variants to complex entity-based models. In our analysis, we

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
nn_coref	69.31	76.23	72.60	55.83	66.07	60.52	54.88	59.41	57.05	63.39
Latent	69.48	76.47	72.81	55.81	65.25	60.16	53.39	60.16	56.58	63.18
Mention-Ent.[18]	68.28	78.42	73.00	55.14	66.73	60.38	50.47	64.54	56.64	63.34

Table 8.29: Comparison with the state-of-the-art system *nn_coref* (Wiseman et al., 2015) on CoNLL-2012 English test data.

concentrated on a detailed comparison of mention pair models and mention ranking models. The mention pair model views coreference resolution as a labeling of pairs of mentions. While this modeling is conceptually simple, it necessitates the use of re-sampling schemes and clustering during post-processing. The mention ranking model improves on the mention pair model by a more adequate modeling of the task. We could attribute most improvements to improved detection of anaphoricity. This may partly be due to our experimental setting: since we work on automatically extracted mentions, most mentions are not anaphoric. Therefore, detection of anaphoricity plays a particularly important role.

The mention ranking model is based on a simple structure and yields very good performance. Therefore, the mention ranking model constitutes a simple, efficient and well-performing baseline.

We could observe further improvements when enriching a variant of a mention ranking model – the antecedent tree model – with entity-based information. However, there were no further improvements when we used more sophisticated inference variants than left-to-right inference. For simplicity and comparison purposes we only used a small entity-based feature set and simple cost functions and heuristics for restraining the search space. Future work should investigate more advanced features, cost functions and heuristics.

[18] Tree-based, using gold roll-in and left-to-right inference.

9 Conclusions

The aim of the research presented in this thesis was to give a treatment of computational coreference resolution that accounts for the structure inherent to the task. In this regard, we identified three major research questions: Is there a *unified structured representation* for approaches to coreference resolution? How can we devise an *error analysis method* that accounts for the structural complexity of the task and employs a useful notion of an error? How do approaches to coreference resolution *differ qualitatively* and to what extent can we attribute these differences to the representations? In this chapter, we revisit the research questions and summarize our contributions towards answering the questions (Section 9.1). Furthermore, we discuss some avenues for future work (Section 9.2).

9.1 Contributions

In this thesis, we considered three main research questions. We now discuss how the research presented in this thesis contributes to answering these research questions.

A Unified Representation for Approaches to Coreference Resolution. We observed that approaches to coreference resolution can be understood as predictors of *latent structures*, which are structures that encode coreference information but are not annotated in the data. We formalized this observation by devising a machine learning framework for coreference resolution. In this framework, a unified representation of approaches to coreference resolution is obtained by casting coreference resolution as *latent structured prediction*. For representing the structures, we employed labeled directed hypergraphs. In our framework, parameters are estimated by using the perceptron algorithm and the learning to search paradigm.

In order to demonstrate the comprehensiveness of our framework we expressed all influential classes of machine learning approaches to coreference resolution in the

framework. In particular, we showed how mention pair models, mention ranking models, antecedent trees, mention-entity models and entity-entity models can be represented. We discussed variants from the literature as well as novel variants. We carved out differences in the representation and in modeling assumptions.

Appropriate Error Analysis. In order to derive an error analysis method for coreference resolution, we first discussed desiderata for such a method: it must cope with the set-based nature of the coreference resolution task, the error representation should be useful and the method should be flexible with respect to the notion of an error. We proposed an error analysis framework that fulfills the desiderata. In this framework, reference and system entities are represented as graphs. Errors are extracted by computing spanning trees of these graphs, which are then compared to partitions of the graphs. In this framework, different notions of an error can be modeled by using different spanning tree algorithms. We discussed various linguistically motivated and data-driven spanning tree algorithms for extracting recall and precision errors.

In-depth Analysis of Qualitative Differences. We made use of the proposed machine learning framework and the error analysis method to perform a large-scale in-depth analysis of coreference resolution approaches on CoNLL-2012 data. We devised an error categorization tailored to our comparison setting and discussed implementations of coreference resolution approaches in order of increasing complexity and expressiveness. In our analysis, we compared approaches according to the structure they are based on and according to a variety of parameters, including cost functions, training data resampling and pruning, and substructure factorization.

We assessed the impact and contribution of variations in the structure and in the parameters. In particular, we found that a mention ranking architecture performs very well when using a suitable cost function. Using the error analysis method and insights obtained from the representation of the approach in our framework we could attribute the improvements to the more adequate structure used by the mention ranking approach. The structure used allows for an improved modeling of anaphoricity and for slightly better antecedent selection. We could observe further improvements for a tree-based mention-entity model with left-to-right inference.

9.2 Future Work

Based on the research presented in this thesis, several avenues for future work exist, both within coreference resolution and within other natural language processing tasks. We discuss three possible extension of the work presented in this thesis.

Improving Entity-centric Approaches. While we showed how to express entity-centric approaches in our framework, the performance of most of the implemented models was not satisfactory. For simplicity and efficiency, we only considered a small set of entity-based features and employed only local, edge-factoring cost functions that do not require roll-outs during learning to search. We believe that entity-centric models can benefit from employing more sophisticated features and cost functions. Some work on coreference resolution uses more sophisticated cost functions (Stoyanov and Eisner, 2012; Ma et al., 2014; Clark and Manning, 2015), but does not compare to simpler cost functions. Future work should evaluate the contribution of adding more advanced features and cost functions to the entity-centric models in our framework.

Furthermore, in order to perform efficient inference, we had to heuristically constrain the search space of models that have complicated inference schemes. As this may negatively affect performance, future work should investigate more efficient implementations and more appropriate methods to constrain the search space.

Knowledge for Coreference Resolution. As we have discussed in the introduction, two main challenges for coreference resolution are the variation in knowledge sources required to resolve coreference and the inherent structural complexity of the task. While we only considered the second challenge in this thesis, the methods and insights presented here are also applicable to work on the first challenge.

As a starting point for such work, the error analysis methods can again be applied to the models presented in this thesis. However, for work on the first challenge, the analysis should be performed with a different focus: How can the errors be traced back to missing or erroneous knowledge? How do the models differ in handling the knowledge which is already provided to the models? Based on such an analysis, one can quantify and assess the contribution of the features and the potential contribution of knowledge sources that are not yet included, such as knowledge bases like YAGO (Hoffart et al., 2011). Then, relying on the error analysis method and the unified representation, knowledge can be added to the model, and impact, contributions and

problems of the additional knowledge can be analyzed in detail across a wide variety of different models.

Analysis and Representation Frameworks for Other Tasks. The analysis of appropriate error analysis methods and underlying structures can be extended to other natural language processing tasks that lack these methods and/or a unified representation. As an example we discuss entity linking, the task of mapping named entities to the corresponding entries in a knowledge base (Shen et al., 2015).

By definition of the task, the output of entity linking approaches can be represented as a graph, where the nodes consist of entity mentions and knowledge base entries. There is an edge between two mentions if they are mapped to the same entry, and there is an edge between a mention and an entry if the mention is mapped to the entry. This representation can be the basis for investigating two research questions. First, it can be investigated whether a useful notion of error can be extracted from this representation. If this is possible, the notion should be compared to existing work on error analysis for entity linking (Heinzerling and Strube, 2015). Methods for error analysis for entity linking will help practitioners and researchers to improve their system, and they deepen the understanding of the task. Second, one can perform an extensive literature review and analysis to determine how different approaches to entity linking tackle the prediction of this structure. This can lead to a uniform representation, as in coreference resolution, or – if the representation is not uniform – will allow for a clean analysis of the differences between entity linking approaches. As in our work on coreference resolution, the newly devised error analysis methods can be employed to determine qualitative differences between individual approaches.

List of Figures

List of Tables

List of Algorithms

Bibliography

Alfred V. Aho and Stephen C. Johnson. 1974. LR parsing. *ACM Computing Surveys*, 6 (2):99–124.

Chinatsu Aone and Scott W. Bennett. 1996. Applying machine learning to anaphora resolution. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pages 302–314. Springer, Berlin.

Mira Ariel. 1988. Referring and accessibility. *Journal of Linguistics*, 24(1):65–87.

Mira Ariel. 1990. *Accessing Noun Phrase Antecedents*. Routledge, London, U.K.; New York, N.Y.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pages 563–566.

Mohit Bansal and Dan Klein. 2012. Coreference semantics from web features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Jeju Island, Korea, 8–14 July 2012, pages 389–398.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 294–303.

Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics,* Sydney, Australia, 17–21 July 2006, pages 33–40.

Roberto Bisani. 1987. Beam search. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 56–58. John Wiley, New York, N.Y.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 49–55.

Anders Björkelund and Jonas Kuhn. 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),* Baltimore, Md., 22–27 June 2014, pages 47–57.

Anders Björkelund and Pierre Nugues. 2011. Exploring lexicalized features for coreference resolution. In *Proceedings of the Shared Task of the 15th Conference on Computational Natural Language Learning,* Portland, Oreg., 23–24 June 2011, pages 45–50.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data,* Vancouver, B.C., Canada, 10–12 June 2008, pages 1247–1250.

Jie Cai and Michael Strube. 2010a. End-to-end coreference resolution via hypergraph partitioning. In *Proceedings of the 23rd International Conference on Computational Linguistics,* Beijing, China, 23–27 August 2010, pages 143–151.

Jie Cai and Michael Strube. 2010b. Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of the SIGdial 2010 Conference: The 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue,* Tokyo, Japan, 24–25 September 2010, pages 28–36.

Jie Cai, Éva Mújdricza-Maydt, and Michael Strube. 2011. Unrestricted coreference resolution via global hypergraph partitioning. In *Proceedings of the Shared Task of the 15th Conference on Computational Natural Language Learning,* Portland, Oreg., 23–24 June 2011, pages 56–60.

Claire Cardie and Kiri Wagstaff. 1999. Noun phrase coreference as clustering. In *Proceedings of the 1999 SIGDAT Conference on Empirical Methods in Natural Language*

Processing and Very Large Corpora, College Park, Md., 21–22 June 1999, pages 82–89.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Nick Rizzolo, Mark Sammons, and Dan Roth. 2011. Inference protocols for coreference resolution. In *Proceedings of the Shared Task of the 15th Conference on Computational Natural Language Learning,* Portland, Oreg., 23–24 June 2011, pages 40–44.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. Illinois-Coref: The UI system in the CoNLL-2012 shared task. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 113–117.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 601–612.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the 32nd International Conference on Machine Learning,* Lille, France, 6–11 July 2015, pages 2058–2066.

Bin Chen, Jian Su, Sinno Jialin Pan, and Chew Lim Tan. 2011. A unified event coreference resolution by integrating multiple resolvers. In *Proceedings of the 5th International Joint Conference on Natural Language Processing,* Chiang Mai, Thailand, 8–13 November 2011, pages 102–110.

Chen Chen and Vincent Ng. 2013. Linguistically aware coreference evaluation metrics. In *Proceedings of the 6th International Joint Conference on Natural Language Processing,* Nagoya, Japan, 14–18 October 2013, pages 1366–1374.

Nancy Chinchor. 2001. Message Understanding Conference (MUC) 7. LDC2001T02, Philadelphia, Penn: Linguistic Data Consortium.

Nancy Chinchor and Beth Sundheim. 2003. Message Understanding Conference (MUC) 6. LDC2003T13, Philadelphia, Penn: Linguistic Data Consortium.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

David Chrystal. 2008. *A Dictionary of Linguistics and Phonetics*. Blackwell Publishing, Oxford, UK.

Herbert H. Clark and Catherine R. Marshall. 1981. Definite reference and mutual knowledge. In A.K. Joshi, B.L. Webber, and I.A. Sag, editors, *Elements of Discourse Understanding*, pages 10–63. Cambridge University Press, Cambridge, Mass.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 1405–1415.

Kevin Clark and Christopher D. Manning. 2016. 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),* Berlin, Germany, 7–12 August 2016, pages 643–653.

William W. Cohen. 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning,* Lake Tahoe, Cal., pages 115–123.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, Penn.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing,* Philadelphia, Penn., 6–7 July 2002, pages 1–8.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pages 111–118.

Dennis Connolly, John D. Burger, and David S. Day. 1994. A machine learning approach to anaphoric reference. In *Proceedings of the International Conference on New Methods in Language Processing,* Manchester, U.K., 14–16 September 1994, pages 255–261.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Aron Culotta, Michael Wick, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics,* Rochester, N.Y., 22–27 April 2007, pages 81–88.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide. Technical Report ILK 04-02, ILK Tilburg.

Hal Daumé III. 2006. *Practical structured learning techniques for natural language processing*. PhD thesis, University of Southern California.

Hal Daumé III and Daniel Marcu. 2005a. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 97–104.

Hal Daumé III and Daniel Marcu. 2005b. Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of the International Conference on Machine Learning,* Bonn, Germany, 7–11 August 2005, pages 169–176.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75:297–325.

Hal Daumé III, John Langford, and Stéphane Ross. 2014. Efficient programmable learning to search. *arXiv preprint*, abs/1408.1837.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation,* Genoa, Italy, 22–28 May 2006, pages 449–454.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 660–669.

Pascal Denis and Jason Baldridge. March 2009. Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42: 87–96.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014. HC-search: a learning framework for search-based structured prediction. *Journal of Artificial Intelligence Research*, 50:369–407.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121−2159.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 1971–1982.

Greg Durrett, David Hall, and Dan Klein. 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),* Sofia, Bulgaria, 4–9 August 2013, pages 114–124.

Miriam Eckert and Michael Strube. 2000. Dialogue acts, synchronising units and anaphora resolution. *Journal of Semantics*, 17(1):51–89. doi: 10.1093/jos/17.1.51.

Charles Elkan. 2001. The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence,* Seattle, Wash., 4–10 August, 2001, pages 973–978.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 41–48.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics*, 40(4):801–835.

Jenny Rose Finkel and Christopher Manning. 2008. Enforcing transitivity in coreference resolution. In *Companion Volume to the Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics,* Columbus, Ohio, 15–20 June 2008, pages 45–48.

Kari Fraurud. 1990. Definiteness and the processing of noun phrases in natural discourse. *Journal of Semantics*, 7:395–433.

Yoav Freund and Robert Shapire. 1999. Large margin classification using the Perceptron algorithm. *Machine Learning*, 37:277–296.

Peter Geibel and Fritz Wysotzk. 2003. Perceptron based learning with example dependent and noisy costs. In *Proceedings of the 20th International Conference on Machine Learning,* Washington, D.C., 21–24 August 2003, pages 218–225.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of Human Language Technologies 2010: The Conference of the North American Chapter of the Association for Computational Linguistics,* Los Angeles, Cal., 2–4 June 2010, pages 742–750.

Peter C. Gordon, Barbara J. Grosz, and Laura A. Gilliom. 1993. Pronouns, names, and the centering of attention in discourse. *Cognitive Science*, 17:311–347.

Jeanette K. Gundel, Nancy Hedberg, and Ron Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, 69:274–307.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric Bayesian model. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics,* Prague, Czech Republic, 23–30 June 2007, pages 848–855.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity centered model. In *Proceedings of Human Language Technologies 2010: The Conference of the North American Chapter of the Association for Computational Linguistics,* Los Angeles, Cal., 2–4 June 2010, pages 385–393.

Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2013. Latent anaphora resolution for cross-lingual pronoun projection. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 380–391.

Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

Benjamin Heinzerling and Michael Strube. 2015. Visual error analysis for entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics: System Demonstrations,* Beijing, China, 26–31 July 2015, pages 37–42.

Jerry R. Hobbs. 1976. Pronoun resolution. Technical Report 76-1, Dept. of Computer Science, City College, City University of New York.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, and Gerhard Weikum. 2011. YAGO2: Exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th World Wide Web Conference,* Hyderabad, India, 28 March – 1 April, 2011, pages 229–232.

Véronique Hoste. 2005. *Optimization issues in machine learning of coreference resolution*. PhD thesis, Universiteit Antwerpen, Faculteit Letteren en Wijsbegeerte, Antwerpen, Netherlands.

Véronique Hoste and Guy De Pauw. 2006. KNACK-2002: a richly annotated corpus of Dutch written text. In *Proceedings of the 5th International Conference on Language Resources and Evaluation,* Genoa, Italy, 22–28 May 2006, pages 1432–1437.

Yufang Hou, Katja Markert, and Michael Strube. 2013. Global inference for bridging anaphora resolution. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Atlanta, Georgia, 9–14 June 2013, pages 907–917.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Montréal, Québec, Canada, 3–8 June 2012, pages 142–151.

Rodney Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge grammar of the English language*. Cambridge University Press, Cambridge, U.K.

Robert A. Hummel and Steven W. Zucker. 1983. On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(3):267–287.

Manfred Klenner. 2007. Enforcing consistency on coreference sets. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing,* Borovets, Bulgaria, 27–29 September 2007, pages 323–328.

Hamidreza Kobdani, Hinrich Schuetze, Michael Schiehlen, and Hans Kamp. 2011. Bootstrapping coreference resolution using word associations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 783–792.

Varada Kolhatkar and Graeme Hirst. 2012. Resolving "This-issue" anaphora. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 1255–1265.

Joseph Kruskal. 1956. On the shortest spanning subtree and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50.

Jonathan K. Kummerfeld and Dan Klein. 2013. Error-driven analysis of challenges in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 265–277.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Emmanuel Lassalle and Pascal Denis. 2015. Joint anaphoricity detection and coreference resolution with constrained latent structures. In *Proceedings of the 29th Conference on the Advancement of Artificial Intelligence,* Austin, Texas, 25–30 July 2015, pages 2274–2280.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Shared Task of the 15th Conference on Computational Natural Language Learning,* Portland, Oreg., 23–24 June 2011, pages 28–34.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 25–32.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 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,* Barcelona, Spain, 21–26 July 2004, pages 136–143.

Xiaoqiang Luo, Marta Recasens, Sameer Pradhan, and Eduard Hocy. 2014. An extension of BLANC to system mentions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Baltimore, Md., 22–27 June 2014, pages 24–29.

Chao Ma, Janardhan Rao Doppa, John Walker Orr, Prashanth Mannem, Xiaoli Z. Fern, Thomas G. Dietterich, and Prasad Tadepalli. 2014. Prune-and-score: learning for greedy coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 2115–2126.

Sebastian Martschat. 2013. Multigraph clustering for unsupervised coreference resolution. In *51st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Student Research Workshop,* Sofia, Bulgaria, 5–7 August 2013, pages 81–88.

Sebastian Martschat and Michael Strube. 2014. Recall error analysis for coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 2070–2081.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

Sebastian Martschat, Jie Cai, Samuel Broscheit, Éva Mújdricza-Maydt, and Michael Strube. 2012. A multigraph model for coreference resolution. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 100–106.

Sebastian Martschat, Patrick Claus, and Michael Strube. 2015a. Plug latent structures and play coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics: System Demonstrations,* Beijing, China, 26–31 July 2015, pages 61–66.

Sebastian Martschat, Thierry Göckel, and Michael Strube. 2015b. Analyzing and visualizing coreference resolution errors. In *Proceedings of the 2015 Conference of the North Americal Chapter of the Association for Computational Linguistics: Demonstration Session,* Denver, Col., 31 May – 5 June 2015, pages 6–10.

Andrew McCallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web,* Acapulco, Mexico, August 9–10, 2003, pages 79–86.

Joseph F. McCarthy and Wendy G. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence,* Montréal, Canada, 20–25 August 1995, pages 1050–1055.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013),* Lake Tahoe, Nevada, December 5-8, 2013, pages 3111–3119.

Ruslan Mitkov. 2002. *Anaphora Resolution*. London, U.K.: Longman.

Nafise Sadat Moosavi and Michael Strube. 2014. Unsupervised coreference resolution by utilizing the most informative relations. In *Proceedings of the 25th International Conference on Computational Linguistics,* Dublin, Ireland, 23–29 August 2014, pages 644–655.

Thomas S. Morton. 2000. Coreference for NLP applications. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics,* Hong Kong, China, 1–8 August 2000, pages 173–180.

MUC-6. 1995. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann, San Francisco, Cal.

MUC-7. 1998. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Morgan Kaufmann, San Francisco, Cal.

Ann E. Mulkern. 1996. The game of names. In Thorstein Fretheim and Jeanette K. Gundel, editors, *Reference and Referent Accessibility*, pages 481–563. John Benjamins, Amsterdam, The Netherlands.

Vivi Nastase, Michael Strube, Benjamin Börschinger, Cäcilia Zirn, and Anas Elghafari. 2010. WikiNet: A very large scale multi-lingual concept network. In *Proceedings of the 7th International Conference on Language Resources and Evaluation,* La Valetta, Malta, 17–23 May 2010.

Vincent Ng. 2004. Learning noun phrase anaphoricity to improve coreference resolution. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pages 151–158.

Vincent Ng. 2008. Unsupervised models for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 640–649.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* Philadelphia, Penn., 7–12 July 2002, pages 104–111.

Cristina Nicolae and Gabriel Nicolae. 2006. BestCut: A graph algorithm for coreference resolution. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing,* Sydney, Australia, 22–23 July 2006, pages 275–283.

NIST. 2003. ACE – Automatic Content Extraction. http://www.nist.gov/speech/tests/ace/index.htm.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses. An Introduction*. Wiley, New York.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pages 143–150.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics,* New York, N.Y., 4–9 June 2006, pages 192–199.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 650–659.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Shared Task of the 15th Conference on Computational Natural Language Learning,* Portland, Oreg., 23–24 June 2011, pages 1–27.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 1–40.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Baltimore, Md., 22–27 June 2014, pages 30–35.

Robert C. Prim. 1957. Shortest connection networks and some generalisations. *Bell System Technical Journal*, 36:1389–1401.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, Cal.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1991. *A Comprehensive Grammar of the English Language*. Longman, London, U.K.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing,* Singapore, 6–7 August 2009, pages 968–977.

Altaf Rahman and Vincent Ng. 2011a. Narrowing the modeling gap: A cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research*, 40: 469–521.

Altaf Rahman and Vincent Ng. 2011b. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 814–824.

Marta Recasens and Eduard Hovy. 2009. A deeper look into features for coreference resolution. In *Proceedings of the 7th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2009),* Goa, India, 5–6 November 2009, pages 29–42.

Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Frank Rosenblatt. 1958. The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408. (Reprinted in *Neurocomputing* (MIT Press, 1988)).

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011,* Fort Lauderdale, Fl., 11–13 April 2011, pages 627–635.

Emili Sapena, Lluís Padró, and Jordi Turmo. 2013. A constraint-based hypergraph partitioning approach to coreference resolution. *Computational Linguistics*, 39(4): 847–884.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge Data Engineering*, 27(2):443–460.

Yang Song, Jing Jiang, Wayne Xin Zhao, Sujian Li, and Houfeng Wang. 2012. Joing learning for coreference resolution with Markov Logic. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 1245–1254.

Wee Meng Soon, Hwee Tou Ng, and Chung Yong Lim. 1999. Corpus-based learning for noun phrase coreference resolution. In *Proceedings of the 1999 SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora,* College Park, Md., 21–22 June 1999, pages 285–291.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27 (4):521–544.

Josef Steinberger, Massimo Poesio, Mijail A. Kabadjov, and Karel Ježek. 2007. Two uses of anaphora resolution in summarization. *Information Processing and Management*, 43(6):1663–1680.

Mechthild Stoer and Frank Wagner. 1997. A simple Min-cut algorithm. *Journal of the ACM*, 44(4):585–591.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *Proceedings of the 24th International Conference on Computational Linguistics,* Mumbai, India, 8–15 December 2012, pages 2519–2534.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2009a. Reconcile: A coreference resolution research platform. Technical report, Cornell University.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009b. 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 Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing,* Singapore, 2–7 August 2009, pages 656–664.

Michael Strube and Simone Paolo Ponzetto. 2006. WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence,* Boston, Mass., 16–20 July 2006, pages 1419–1424.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of the 21th International Joint Conference on Artificial Intelligence,* Pasadena, Cal., 14–17 July 2009, pages 1236–1242.

Don Tuggener. 2014. Coreference resolution evaluation for higher level applications. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics,* Gothenburg, Sweden, 26–30 April 2014, pages 231–235.

Olga Uryupina. 2007. *Knowledge acquisition for coreference resolution*. PhD thesis, Saarland University, Saarbrücken, Germany.

Olga Uryupina. 2008. Error analysis for learning-based coreference resolution. In *Proceedings of the 6th International Conference on Language Resources and Evaluation,* Marrakech, Morocco, 26 May – 1 June 2008, pages 1914–1919.

Kees van Deemter and Rodger Kibble. 2000. On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics*, 26(4):629–637.

Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Proceedings of the 6th International Conference on Language Resources and Evaluation,* Marrakech, Morocco, 26 May – 1 June 2008.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, Cal. Morgan Kaufmann.

Christopher Walker, Stephanie Strassel, Julie Medero, and Maeda Kazuaki. 2006. ACE 2005 multilingual training corpus. LDC2006T06, Philadelphia, Penn.: Linguistic Data Consortium.

Bonnie L. Webber. 1979. *A formal approach to discourse anaphora*. Garland, New York, N.Y.

Kellie Webster and James R. Curran. 2014. Limited memory incremental coreference resolution. In *Proceedings of the 25th International Conference on Computational Linguistics,* Dublin, Ireland, 23–29 August 2014, pages 2129–2139.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. OntoNotes: A large training corpus for enhanced processing. In J. Olive, C. Christianson, and J. McCary, editors, *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitations*, pages 54–63. Springer, Heidelberg, Germany.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mo-

hammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes release 5.0. LDC2013T19, Philadelphia, Penn.: Linguistic Data Consortium.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 1416–1426.

Sam Wiseman, Alexander M. Rush, and Stuart Shieber. 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,* San Diego, Cal., 12–17 June 2016, pages 994–1004.

Xiaofeng Yang and Jian Su. 2007. Coreference resolution using semantic relatedness information from automatically discovered patterns. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics,* Prague, Czech Republic, 23–30 June 2007, pages 528–535.

Xiaofeng Yang, Guodung Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics,* Sapporo, Japan, 7–12 July 2003, pages 176–183.

Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu, and Sheng Li. 2008. An entity-mention model for coreference resolution with Inductive Logic Programming. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies,* Columbus, Ohio, 15–20 June 2008, pages 843–851.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proceedings of the 26th International Conference on Machine Learning,* Montréal, Québec, Canada, 14–18 June 2009, pages 1169–1176.

Desislava Zhekova. 2011. Instance sampling for multilingual coreference resolution. In *Proceedings of the Student Research Workshop associated with RANLP 2011,* Hissar, Bulgaria, September 13, 2011, pages 150–155.

A Additional Results on Test Data

In this appendix, we present results for all models discussed in Chapter 8 on CoNLL-2012 test data.

A.1 Mention Pair Models

Tables A.1 and A.2 show the results of different mention pair resampling and substructure factorization variants on CoNLL-2012 test data.

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Vanilla pair										
All pairs	66.43	77.06	**71.35**	54.77	51.52	53.09	34.48	**61.22**	44.12	56.19
Soon	**80.24**	47.82	59.93	**74.17**	18.99	30.25	25.54	33.51	28.99	39.72
Mod. Soon	72.78	68.93	70.80	62.67	46.40	52.32	43.06	51.88	47.06	57.06
Closest first										
All pairs	62.24	78.26	69.34	45.07	64.38	53.02	41.49	57.89	48.34	56.90
Soon	73.71	50.63	60.02	61.72	35.40	44.99	47.37	38.06	42.21	49.07
Mod. Soon	68.38	70.44	69.40	52.74	58.65	55.54	**52.56**	50.70	**51.61**	**58.85**
Best first										
All pairs	62.81	**78.65**	69.84	47.32	**64.56**	54.61	41.11	59.41	48.59	57.68
Soon	72.87	50.05	59.34	61.50	34.71	44.37	44.50	37.36	40.62	48.11
Mod. Soon	67.16	71.63	69.32	52.30	60.40	**56.06**	50.61	51.20	50.90	58.76

Table A.1: Results of mention pair models with different resampling variants and `obtain_coreference` instantiations on CoNLL-2012 English test data. Highest values for each column are marked bold.

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Baseline	**68.38**	70.44	69.40	**52.74**	58.65	55.54	52.56	50.70	51.61	58.85
Per Anaphor	67.41	71.29	69.30	50.99	60.51	55.35	**52.78**	50.75	**51.75**	58.80
Per Document	66.01	**74.13**	**69.83**	49.75	**63.74**	**55.88**	50.30	**53.23**	51.73	**59.15**

Table A.2: Results of mention pair models with substructure factorization variants on CoNLL-2012 English test data. The baseline considers each pair as a substructure and employs closest-first clustering and the modified Soon resampling scheme. Highest values for each column are marked bold.

A.2 Mention Ranking and Antecedent Trees

Table A.3 shows results of ranking variants on CoNLL-2012 test data. In addition to the results presented in Section 8.5, the results of mention ranking with the modified Soon resampling scheme and the results of the graph-based models are shown.

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Best first										
Mod. Soon	67.16	71.63	69.32	52.30	60.40	56.06	50.61	51.20	50.90	58.76
Ranking										
Mod. Soon	**73.16**	63.41	67.94	**62.08**	48.11	52.92	50.05	49.05	49.55	56.80
All antec.	71.61	68.55	70.05	58.60	56.74	57.65	**56.23**	51.70	53.87	60.52
No cost	62.91	**81.23**	70.91	46.89	**72.96**	57.09	48.67	58.99	53.34	60.45
Cost	69.62	76.26	72.79	56.10	63.43	59.54	51.97	59.61	55.53	62.62
Latent	69.48	76.47	**72.81**	55.81	65.25	**60.16**	53.39	**60.16**	**56.58**	**63.18**
Antec. Tree	68.63	77.27	72.69	54.83	66.38	60.05	52.63	60.05	56.10	62.95
Graph (Anaph.)	70.94	73.78	72.34	59.85	54.24	56.91	46.34	59.35	52.04	60.43
Graph (Doc)	59.89	80.32	68.62	45.36	64.07	53.11	35.20	60.03	44.38	55.37

Table A.3: Results of ranking variants on CoNLL-2012 English test data. Highest values for each column are marked bold.

A.3 Entity-based Models

Table A.4 shows results of entity-based models on CoNLL-2012 test data. In Section 8.5, we only reported results for the tree-based mention-entity model with left-to-right inference.

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Avg. F_1**
Antec. Tree	68.63	77.27	72.69	54.83	66.38	60.05	52.63	60.05	56.10	62.95
Mention-Entity	(Tree, left-to-right)									
Lrnd. roll-in	68.27	77.40	72.55	54.18	67.36	60.06	**53.03**	59.47	56.07	62.89
Gold roll-in	68.28	78.42	**73.00**	55.14	66.73	60.38	50.47	64.54	**56.64**	**63.34**
Mention-Entity	(Tree, easy-first)									
Lrnd. roll-in	68.43	77.49	72.68	54.06	67.15	59.90	52.91	60.32	56.37	62.98
Gold roll-in	68.69	77.68	72.91	55.16	66.77	**60.41**	51.56	62.23	56.40	63.24
Entity-Entity	(Tree)									
Lrnd. roll-in	68.27	77.31	72.51	54.14	66.72	59.77	52.47	60.45	56.18	62.82
Gold roll-in	**68.94**	77.41	72.93	**55.69**	65.05	60.01	50.39	64.06	56.41	63.12
Mention-Entity	(Hypergraph, left-to-right)									
No cost	63.50	**81.41**	71.35	48.96	70.27	57.71	45.44	**65.72**	53.73	60.93
Hyper cost	63.76	77.57	69.99	46.47	**70.50**	56.02	52.06	57.83	54.79	60.27
Pair cost	67.09	78.12	72.19	52.94	65.45	58.53	48.62	64.66	55.50	62.07

Table A.4: Results of entity-based models on CoNLL-2012 test data. Highest values for each column are marked bold.