Structured learning with latent trees: a joint approach to coreference resolution

Emmanuel Lassalle

Université Paris Diderot (Paris 7)
École Doctorale de Sciences Mathématiques de Paris-Centre 386

Doctorat d'informatique

STRUCTURED LEARNING WITH LATENT TREES: A JOINT APPROACH TO COREFERENCE RESOLUTION

Emmanuel Lassalle

Mai 2015

Thèse sous la direction de :
Laurence Danlos et Pascal Denis

Composition du jury :
Laurence Danlos, directrice de thèse
Pascal Denis, co-directeur de thèse
Dan Roth, rapporteur
Liva Ralaivola, rapporteur
Isabelle Tellier, examinatrice

Abstract

This thesis explores ways to define automated coreference resolution systems by using structured machine learning techniques. We design supervised models that learn to build coreference clusters from raw text: our main objective is to get model able to process documents globally, in a structured fashion, to ensure coherent outputs. Our models are trained and evaluated on the English part of the CoNLL-2012 Shared Task annotated corpus with standard metrics. We carry out detailed comparisons of different settings so as to refine our models and design a complete end-to-end coreference resolver.

Specifically, we first carry out a preliminary work on improving the way features are employed by linear models for classification: we extend existing work on separating different types of mention pairs to define more accurate classifiers of coreference links. We then define various structured models based on latent trees to learn to build clusters globally, and not only from the predictions of a mention pair classifier. We study different latent representations (various shapes and sparsity) and show empirically that the best suited structure is some restricted class of trees related to the *best-first* rule for selecting coreference links. We further improve this latent representation by integrating anaphoricity modelling jointly with coreference, designing a global (structured at the document level) and joint model outperforming existing models on gold mentions evaluation. We finally design a complete end-to-end resolver and evaluate the improvement obtained by our new models on detected mentions, a more realistic setting for coreference resolution.

Acknowledgements

First of all I would like to thank Pascal Denis and Laurence Danlos for guiding me during this long journey. During these years, Pascal's intellectual rigour, his high level of requirement but also his constant support have been a great source of motivation to carry out this work. Our long discussions about models and implementation improvements have been decisive for orienting this research and I am grateful for all his attention. Of course, I would have never been able to make it without Laurence's encouragements and patience. I am glad she accepted to supervise my thesis and welcomed me in Alpage research group where I had the best environment to get started on NLP research. Her guidance and her encouragements have been the key to complete my thesis.

I am very grateful to Dan Roth for accepting to be part of the jury, this is a great honour. I also cannot be more thankful to Liva Ralaivola and Isabelle Tellier who agreed to be in the jury and made thoughtful comments when reviewing my thesis in details.

Being part of Alpage research team was a real pleasure, and I am thankful for all its members, who at some point gave me good advice or stimulated my curiosity on various topics: Benoît Crabbé, Marie Candito, Éric de La Clergerie, Djamé Seddah. I feel particularly indebted to Benoît Sagot for his meaningful advises.

We had a great time with the fellow *doctorants*, and I now realize that all these years have passed so quickly. I would like to thank everyone who was once sitting in our small office, and with whom I shared a piece of this adventure: Chloé, Marion, Pierre, Valérie, Luc, Charlotte, Enrique, Juliette, Rosa, Corentin, Marianne. I would also like to greet our post-doc neighbours Damien and Yves. My apologies to those I might have forgotten.

Finally, a big thank you to my family. They have always been here to support me, and to give me courage in the difficult moments. I dedicate my thesis to my wonderful wife, without whom I could not have made it.

Contents

Chapter 1

Introduction

Among the many tasks of Natural Language Processing (NLP), coreference resolution has a fundamental importance for language interpretation. In fact, detecting properly to which entity linguistic expressions in the text refer to (in particular, detecting if two expressions refer to the same object) is crucial to build accurate semantic representations that can serve as a basis to address many other NLP related problems such as information extraction, information retrieval, question-answering systems, automatic summarization or translation. All of these applications can benefit from an accurate computation of references. Conversely, automated coreference resolution depends on the quality of many other tasks in language processing: among others, POS tagging, syntactic parsing, named entity recognition and shallow semantic extraction such as entity typing are part of the inputs of state of the art automated systems. Previous work on the subject has shown that the quality of coreference resolution systems depends on morphology processing like detecting gender and number, syntactic parsing (constituents or dependencies), and semantic knowledge such as lexical knowledge or encyclopedic knowledge Soon *et al.* (2001); Ng & Cardie (2002b); Bengtson & Roth (2008); Recasens & Hovy (2009). Indeed, world knowledge can be required to be able to correctly resolve some anaphora, making the problem of coreference resolution as difficult as understanding the whole semantics of a discourse. Although automated coreference resolvers are not intended to address the complete range of referential expressions, today coreference resolution can be assessed on standard annotated corpora (with a clear range of annotated phenomena) using a variety of standard metrics. More precisely, most annotations concern noun phrase (NP) coreference. NPs are syntactic structures that encompass proper names, common nouns and pronouns, which are the most common referential expressions, i.e. segments of text introducing or pointing to discourse objects (terminology will be clarified after). And because the field had time to mature during the last decades, giving birth to many different empirical approaches to the problem, the challenge has become harder and now requires more advanced models. Today, both linguistic and computational constraints on coreference resolution tend to be integrated in automated systems, which resulted in more sophisticated models than before.

In example 1.1[1], we give a glimpse of how complex can coreference resolution be. Numbered brackets represent part of the text which refer to the same object or event. We can already ask many questions: should we or not annotate coreference copulas (e.g., should we annotate a link between the two parts of “Johann van Beethoven was a harsh instructor”)?

[1] http://en.wikipedia.org/wiki/Ludwig_van_Beethoven

Should coreference links be subject to a context (for instance, transitivity cannot apply though we create links between “the child Beethoven” “Ludwig” and between “Ludwig” and “the old Beethoven")? Should noun phrases referring to an object of the world be annotated even if they are not coreferring with other parts of the text (so that we would make a difference between them and non referential noun phrases)? With a simple example, we see how complex it can be to define the problem of resolving coreference. We will provide more detailed examples in Chapter 2.

[[Beethoven]$_1$'s first music teacher]$_2$ was **[[his]$_1$ father]$_2$**. Although tradition has it that **[[Johann van Beethoven]$_2$ was a harsh instructor, and that [[the child Beethoven]$_1$, "made to stand at the keyboard, was often in tears,"]$_3$** the Grove Dictionary of Music and Musicians claimed that no solid documentation supported **[this]$_3$**, and asserted that "speculation and myth-making have both been productive."

Figure 1.1: A first example, with some coreference annotations

Anaphora, defined in linguistics as a pattern consisting of an expression which can only be interpreted through its context, has been addressed in formal models early in the history of NLP (Webber, 1978; Hirst, 1981). Among other objectives, this kind of work intended to define under which conditions anaphoric references were available in a text. During the same period, the first pronoun resolution systems emerged (Hobbs, 1978; Brennan *et al.*, 1987; Rich & LuperFoy, 1988; Carbonell & Brown, 1988). These were defined through a combination of formal hand-crafted rules and had a rather restrained scope, but they are still relevant today, especially since more accurate syntactic parsers have become available. Anaphora has also been (and continues to be) addressed by semantic theories of discourse, enlightening empirical approaches and providing new tools to tackle the problem (Kamp & Reyle, 1993; Asher & Lascarides, 2003; Lascarides & Asher, 2007). Apart from that, it is worth mentioning that grammar theory also contributed to the study of anaphora (for instance, see Grosz *et al.* (1995); Walker *et al.* (1998)), providing meaningful constraints which are now partly modelled by the features of many automated coreference resolvers.

However, the limits of hand-crafted rule-based models rapidly appear when trying to process large volumes of texts, from diverse domains. Besides the question of language and domain adaptation or required world knowledge to address coreference resolution in its generality, these models are quite limited in the sense that their imperfections can only be corrected through additional hand-crafted patterns, exceptions to the rules, exceptions to the exceptions and so on. This kind of rule-based methods is not limited to coreference resolution, there are plenty of them for coping with NLP related problems, and one popular approach to avoid overloading a system with hard-coded rules is to rely on machine learning techniques, many of them employing high-dimensional statistical models. In fact, the availability of annotated corpora of good quality made it possible to create efficient data-driven automated systems for coreference resolution. By modelling the task of finding coreference links as a classification problem in high dimension, it is then possible to avoid over-specifying a set of rules (which can result in a lot of time spent in dealing with exceptions, exceptions to the exceptions and so on) and let the model learn from data.

Machine learning approaches to coreference resolution came out in the early 2000s with supervised models (Soon *et al.*, 2001; Ng, 2003). These supervised models learn to retrieve the same kind of coreference links as those observed on an annotated corpus. Since then, it has become the mainstream approach to the problem. The models have been improved thanks to the combination of new machine learning techniques and progress in computational linguistics (especially on studying the factors implied in coreference, and how to express them through the features – or dimensions – of the model). Many advances have been made with joint models on the one hand (Culotta *et al.*, 2007; Denis & Baldridge, 2007) and unsupervised models on the other hand (Haghighi & Klein, 2007; Poon & Domingos, 2008; Ng, 2008)). Nevertheless, supervised models are currently those providing the best results. Recently, Stanford's multi-pass sieves (Lee *et al.*, 2011), a fully deterministic system based on a clever ordering of resolution patterns and the propagation of constraints to make as precise decisions as possible on defining coreference links or not, obtained very good and somehow unexpected results. Even though this particular architecture challenged all data driven approaches at that time, subsequent improvements on the sieves obtained by combining them with statistical models showed that their robustness could be enhanced by machine learning, providing even better results (Chen & Ng, 2012; Ratinov & Roth, 2012).

Today, the range of empirical approaches to coreference resolution is pretty large[2], and many directions for research are possible. In this thesis, we work in the machine learning framework, and give ourselves a challenge which is threefold:

- First, we will carry out a preliminary work on the features employed in the statistical models and find a way to improve the representation of data which is coherent from a linguistic point of view. This part is essential because the performance of our automated system will depend on the quality of data representation. Data representation primarily concerns the way we extract features from data, that is, all the preprocessing modules in an automated system. This preprocessing must be as accurate as possible to avoid propagating errors in the whole system. Another aspect, which is related to the linear model we will use downstream, is to improve the vector representation of data.

- The second line of work is to thoroughly study how to represent a document with a single structure, which helps addressing coreference resolution with a global coherence. Indeed, coreference is a global phenomenon, and coreference links span all over the document, with structural constraints such as transitivity of the relation, or incompatibility between certain types of expressions. However, computing a global structure can be a challenging task, because one has to find a good trade-off between complexity (computation should be fast, for instance, at most quadratic in size of the document) and expressiveness (to be useful, a structure should provide more accurate results than local models).

[2]Research is more oriented towards developing coreference resolvers and evaluating them on annotated data. See Chapter 4 about the state of the art in automated coreference resolution.

- Finally, our third objective is to process coreference-related tasks jointly with coreference resolution. A joint approach to these tasks would help to reduce propagation of errors, which are difficult to detect and revise in a pipeline structure, and potentially improve the quality of their resolution.

A complete outline of this thesis in given in Section 1.3, at the end of this chapter. Before that, let us underline the importance of designing accurate automated coreference resolvers by detailing some of the applications of this NLP module.

1.1 Current issues in coreference resolution

As already mentioned, coreference resolution has an important status in natural language processing and its applications. Having a good partitioning of mentions can help building a semantic representation of the text. This is quite an idealistic view since it is difficult to consider coreference resolution as task independent from other semantic analysis. A perfect coreference resolution should in practice involve complex mechanisms which in turn could be used themselves to build the semantic representation. In fact, as we can see in 2.1, complex mechanisms can underlie coreference and in certain cases require external knowledge (i.e. common knowledge not introduced in the document) to be understood. Besides, coreference can find many applications in NLP-related tasks such as information extraction or retrieval (it is easy to imagine that just pronoun resolution can help retrieving more accurate information), automated question-answering, machine translation, etc.

However, today some authors argue that it is perhaps too early to evaluate the benefits of using coreference resolver to improve other NLP tasks: using their solver on a bench of experiments on text summarization, term extraction and text categorization, Mitkov *et al.* (2007) concluded that “the deployment of anaphora has a positive but rather limited impact". This study is a bit outdated, but there are many possible explanations for this limited impact of coreference resolution: coreference resolvers may not be accurate enough today to help other NLP tasks. But this could be due to another reason (also raised by Mitkov *et al.* (2007)): integrating coreference resolution as a single input to other systems may result in too many errors propagating, suggesting that a joint approach to these tasks along with coreference resolution could result in real improvement of the tasks. In fact, with a few experiments on relation extraction and entity typing, Chan & Roth (2010) show that a joint approach is more beneficial too these tasks altogether than addressing them separatly. Apart from that, a remarkable domain of application of coreference is the automated analysis of medical reports, where coreference can play an important role (Zhou *et al.*, 2006; Zheng *et al.*, 2011; Uzuner *et al.*, 2012). In that domain, the need of automated processing of reports is crucial because of the accelerating production of large amount of texts.

Today there are several trends in the development of automated coreference resolver. Because of the availability of annotated corpora of good quality in some languages, most sys-

tems are based on supervised machine learning models and one of the challenges is to design a model that takes into account both the linguistic properties we know about anaphora and the computational constraints (in particular, their algorithmic complexity) of the models we use. The whole matter is to find such a well balanced model. As written at the beginning of this chapter, among the main issues, finding relevant features for a statistical model is always an objective one can work on when relying on machine learning techniques. Progress on this side have been made since the beginning of statistical coreference resolvers and work is still going on. Knowledge extraction (like ontology construction from text) is one option that could help adding more features (e.g. name entity typing) needed when common knowledge is required to infer coreference.

Another modeling problem is to design systems that are able to process coreference globally, making a single coherent decision on coreference links instead of many independent local decisions, which has been the mainstream approach for long. As mentioned above, transitivity is part of the properties of coreference, and any two coreference links should come with a third one. This has been tackled with different tools or cluster representations (e.g., set instead of graphs), some of them being limited by their computational cost like enforcing constraints with integer linear programming (Denis & Baldridge, 2007). It is also important to define models that are scalable in the sense that they are supposed to process large amount of text (potentially with time constraints). For instance, integer linear programming makes it possible to integrate constraints easily, but the resulting combinatoric problem to solve falls in the class of NP-complete problems, and is typically a way we would try to avoid.

Finally, an additional challenge on coreference resolution is to design models that can be extended efficiently to the joint resolution of coreference along with other related tasks. Indeed joint approaches can potentially reduce the propagation of errors, and help the model make better decisions. It can also be a good way to fully integrate coreference resolution for improving other tasks instead of having a pipeline architecture restraining a lot the communication between different NLP modules[3].

1.2 Summary of the main contributions

We end this introductory chapter by listing our contributions to the domain of coreference resolution. These are separated in Chapters 5, 6, 7 and 8. Firstly, we define a new method for enhancing the representation of mention pairs in pairwise models, which is linguistically justified and computationally efficient. It allows us to define a pairwise model competing with similar models, but also it can be employed in other linear prediction methods such as the structured models we develop later on. Compared to other feature selection methods, ours relies on a linguistic approach and extends in some ways the approach of Denis & Baldridge (2008), by separating different kinds of coreference links and processing them separately (i.e., through

[3]See chapter 8 for a detailed comparison of pipeline vs joint architecture.

separated sets of features). Our optimization algorithm explores a large space of representations by using dynamic programming to find the best one with regards to the F1-score of pair classification.

Secondly, we carry out a detailed study on latent tree structures that we use for representing clusters and defining a global approach to coreference resolution. These structures have been studied previously but separately by other authors. By both topological and quantitative considerations, we find a better structure which can easily be used jointly with strict constraints on mention pairs. This work also makes it possible to compare different kinds of structures in a common framework (linear prediction) and with common features. Indeed, it was not clear before whether the gain obtained by such structures was due to the structure itself, the features of the model, or the learning algorithm. Our experiments quantify the gain to structured approaches and show that "best-first" trees are well suited to coreference resolution compared to other structures.

Thirdly, we extend the latent tree structures as trees representing both coreference links and discourse-new mentions in order to achieve both global and joint learning of coreference resolution along with anaphoricity detection. In fact anaphoricity (or knowing properties such as discourse-new / discourse-old [4]) provides strong constraints on coreference resolution by requiring or invalidating coreference links. A joint approach makes it possible to avoid propagation of errors and achieves very good results in both of these tasks.

Fourthly, we address the problem of resolving coreference on detected mentions. We first improve the state of the art on singleton detection[5]. We then apply our system to detected mentions, and show that increments due to our improvements of the models also appear in this case.

1.3 Dissertation outline

The next three chapters are dedicated to introducing the problem of coreference resolution from an empirical perspective:

- In Chapter 2, we start by introducing the terminology that will be employed throughout the thesis (Section 2.2.1). The vocabulary we employ is standard in the NLP world (even if not always accurate from the point of view of linguistics). We then provide a detailed description of the coreference-related tasks we will address after, and exhibit the interactions between them (Section 2.2.2). The standard metrics for evaluating the quality of coreference clusters are described in Section 2.3. After briefly introducing previous coreference corpora on English, we provide a detailed presentation of the CoNLL-2011

[4] See Section in Chapter for an explanation of this terminology.

[5] Evaluating coreference resolution on gold mentions can be seen as having an oracle for detecting singleton. This scenario becomes less unrealistic when singleton detection with a model is accurate enough.

and 2012 corpora, on which we will carry out all our experiments. We intend to give indications about the impact of ignoring verbal mentions on the evaluation metrics (which were introduced for the first time in the CoNLL-2011 corpus). We underline some of the problems related to the annotation scheme of CoNLL-2011 and 2012 in Section 2.5, and we give our standpoint on the evaluation of coreference resolvers on gold mentions compared to the evaluation on detected mentions.

- Chapter 3 is an introduction to the machine learning techniques employed in this thesis, which are state of the art in NLP. We also give some pointers to learn about other statistical methods used in coreference resolution. Feature representation of data and selection of the features is presented in Section 3.2. Linear classification models are presented in Section 3.3. Online algorithms for training these models are fully described in Section 3.4. We then have a discussion about how to modify these linear predictors and their learning algorithm to achieve structure prediction in Section 3.6.

- Chapter 4 provides a brief history of coreference resolution, showing the evolution of research on the matter, and the orientation toward empirical approaches (see Section 4.1). We next draw up a complete state of the art on coreference in Section 4.2. Because the research trends are diverse on the subject and because coreference resolution is connected to various other tasks in NLP, we preferred to make a thematic presentation of recent work and group articles having a similar approach together instead of just relying on a time line.

The following chapters expose our work and our contributions on coreference resolution, following the three objectives we have defined above:

- In Chapter 5, we propose a new method for significantly improving the performance of pairwise coreference models using linear predictors. Our approach finds feature space for separating coreferential and non coreferential mention pairs optimizing the F1-score, by using combination of features in a similar way to polynomial kernels, but keeping the learning procedure linear. By relying on a dynamic programming technique, the procedure remains tractable and the space is selected efficiently among an exponential number of possibilities. It can be applied to generate a better feature space for a linear classifier and then plugged into more complex, structured model. One thing worth to notice is that this method is linguistically justified and is not just a combinatorial trick to enhance feature spaces.

- Chapter 6 addresses the problem of defining *global* models for resolving coreference. We compare several methods for learning latent structures representing coreference clusters which can be inserted in a framework where we also put accurate *must-link* and *cannot-link* constraints on the pairs of mentions. We study the relationship between standard decoding strategies used with pairwise models and the latent structures, providing both topological properties and quantitative results.

- In Chapter 7 we design a new structured model performing anaphoricity detection and coreference resolution jointly. This structure is based on a latent tree to represent the full coreference and anaphoric structure of a document at a global level. We integrate the constraints studied in Chapter 6 to obtain a robust model that performs both global resolution of coreference and joint processing of the strongly related task that anaphoricity detection is, avoiding propagation of error like in a pipeline model.

- In Chapter 8, we start by improving the state of the art in singleton detection by enhancing the feature set currently employed for this task. We then define a complete architecture for end-to-end coreference resolution, which integrates all the models we defined before. We show that improvement is achieved in this more realistic setting by our joint and structured models.

Chapter 2

The task(s) of Coreference Resolution

In this chapter, we start by describing the technical problems we face when we want to design an automated coreference resolution system. Indeed, the computational tasks depend on the modelling choices that have been made upstream, and on the methodology chosen to address the problem. For instance, anaphoricity determination can be done up front or jointly while resolving coreference. We also examine the dependencies of these task and how errors propagate according to the architecture. We next introduce the evaluation metrics currently used to quantify the accuracy of coreference resolver. Knowing what the metrics express is important, both when designing a resolver, and when analysing its output. Finally we list annotated corpus currently used to rank coreference. Our main focus is on English corpora (all our experiments are in English), and especially on the more recent corpus, published during CoNLL-2012 Shared Task, which contains a large number of texts in diverse domains, and with many different kinds of coreference links annotated.

2.1 Coreference resolution: a linguistic introduction

Before proceeding with computational consideration about the problem of coreference resolution, we need to identify precisely the linguistic phenomenon we will be working on. This section introduces the linguistic context and basic terminology about coreference resolution. The expression "coreference resolution" should be put in perspective with "anaphora resolution" which, as we shall see below, provides a more general framework to view coreference resolution. Although one phenomenon is not subsumed by the other, adopting the anaphora point of view will help us defining the limits of our study, and showing some of the connections which can be tackled in future work. In particular, we will see that there are certain kinds of anaphora involving more complicated semantic relations than coreference and requiring more specific approaches. We will also provide a few examples to illustrate the different constraints and factors that are at stake in the cognitive process of resolving anaphora, and which are modelled in computer systems for solving the task.

2.1.1 Coreference resolution and anaphora resolution

The linguistic literature about anaphora is abundant (for detailed introductions or treatment of the topic, see Webber (1978); Hirst (1981); Mitkov (2002); Poesio *et al.* (2011), and the term **anaphora resolution** was previously employed to designate the process of finding an antecedent to an anaphora (as explained below), but the Message Understanding Conferences (MUC) popularized the term **coreference resolution**, defined as the task of identifying referring expressions and grouping together those which refer to the same discourse entity. We will emphasize some differences between the two tasks and report when they coincide. The MUC conferences were initiated to boost research in information extraction (IE), providing data to evaluate IE systems. Coreference resolution was introduced in MUC-6 (muc, 1995) and MUC-7 (Chinchor, 1998) corpora, illustrating the fact that it is since then considered as a fundamental component of IE systems.

Before going deeper in the linguistic aspects, we shall introduce additional terminology. **Mentions** (also previously called markables) are text or speech constituents referring to a **discourse entity** (or simply entity) which can be, for instance, a person, an object or a place[1], but also an event[2], which is more difficult to apprehend. Discourse entities can be seen as objects under discussion in the text, but not necessarily objects of the world, which is typically the case when dealing with hypothetical situations, as illustrated in the sentence below (Partee, 1973):

> If I had a **[hammer]** I would use **[it]** to break your head.

Discourse elements in a text (or a speech) are not inserted independently and some referential expressions depend on others to be interpreted correctly. This organization of the discourse, also called **cohesion**, provides a more compact way to transmit information. For instance, some mentions have backward dependencies, in the sense that we need to look at the left context (i.e., what comes before) and find a previously mentioned entity to know what they refer to: these mentions are called **anaphora**. The typical example of anaphora are pronouns: they carry almost no semantic information (except their gender and number, and sometimes an entity type like “person") and it requires to find an antecedent to be able to understand the sentence they are part of (with the exception of non-referential “it" pronouns). A simple example of pronominal anaphora can be found in the following example:

> **[The man]** left his office but **[he]** forgot to lock the door.

Anaphora provide an economical mechanism for producing discourse and transmitting information whilst avoiding repetitions when referring several times to the same entity, however they require an additional cognitive process for being fully understood. Indeed, for each anaphora, it is necessary to find a mention to which it points back. This particular mention is

[1] Or more generally, other types of named entities such as organization, date, or some domain specific types (e.g., in medical texts).

[2] Coreference of events are annotated in the recent CoNLL-2011 and 2012 corpora.

also called the **antecedent**: finding the antecedent, or one of the acceptable antecedents of anaphora in its context is the task called **anaphora resolution**. When both the anaphora and the antecedent refer to the same entity, they are said **coreferential** (this is the case in the example above). From now, we can see that there is an asymmetric relation between anaphora and their antecedent, and the complete anaphora resolution task on a given document is to identify all anaphora and resolve them. Anaphora identification (also anaphora detection) is quite easy to achieve if we are mainly interested by pronoun resolution (only pleonastic, or non-referential pronouns are not anaphora and must be eliminated), but it can be more challenging when trying to determine anaphoricity of definite expressions (e.g. “the man"). Some expressions behaving like anaphora but pointing to a “right-context" (i.e. after in the text) are called cataphora. They are not captured by the anaphora framework, but are very similar, though. Finally, when extending the task to resolving anaphora involving different semantic relations than identity (these are called **bridging anaphora** – see 2.1.3), the whole task of identifying anaphora and resolving them becomes much harder.

Anaphora is a fundamental component of discourse: a correct understanding of the logical structure of a text cannot be achieved without resolving anaphora. However, in some cases, resolving anaphora necessitates some general world knowledge and also understanding the discourse. For example, in the following sentences, one should resolve the pronominal anaphora correctly to fully understand what is happening, but to do so, it is required to infer that “Peter” scratched the CD:

> My father gave me the CD that I lent to **[Peter]**. I am really angry with **[him]** now because it is all scratched.

On the other hand, we can address a similar problem: if we look at all entity mentions in the text and try to group together mentions referring to the same entity, we can then define a partition of the mentions. The subsets of coreferent mentions are also called **coreference chains**, and the problem consisting in identifying mentions in the text and building coreference chains is called **coreference resolution**. From a computational point of view, it can be viewed as a clustering task, as the purpose is to build a separation of mentions into equivalence classes. Indeed, coreference is defined as an equivalence relation on mentions contrarily to the asymmetric relation that binds an anaphor to an antecedent in its context (van Deemter & Kibble, 2000). For example the proper name *Michele Obama* and the definite noun phrase *The First Lady of the US*, which are unambiguously defined without looking at their context (one cannot be anaphor and the other its antecedent), do corefer provided we are in the temporal context of nowadays. But finding the coreference relation of the two expressions however requires to have some up-to-date common or encyclopedic knowledge, which make the task even more difficult to be fully addressed.

To sum up, if anaphora and coreference resolutions have some mechanisms in common, especially for finding the reference of pronouns, they also have significant differences in that coreference can happen without depending on the context and that, in general, anaphora can have other semantic relations than coreference with their antecedent. In practice, the techniques

for coping with one resolution or another interleave, and coreferential anaphora resolution technique can be applied to coreference resolution. The empirical methods we will develop in this thesis are dedicated to coreference resolution.

2.1.2 Constraints, preferences and common knowledge

Discourse models from formal linguistics provide a more accurate description of the dependency on context of anaphora (Karttunen, 1969; Webber, 1978; Kamp, 1981), in which a model of interpretation of discourse is dynamically built in the order of the text. In particular, the Discourse Representation Theory (DRT) offers a suitable framework to study the semantic of anaphora. Since our interest is in automated computational models for solving anaphora, we will not go into further details about these theories. However, we must point out the fact that some anaphora resolution systems use such dynamic representations, such as the *Boxer* software (Bos, 2008). Other systems have simplified but similar representations. For instance, with entity ranking models, the dynamic structure that is built along the text is only the clusters of coreferent mentions (Denis & Baldridge, 2008; Rahman & Ng, 2011): every time a new mention is encountered, it is linked to the most satisfying partial cluster previously built or put in a new cluster if the model considers it does not belong to any previous cluster.

The mechanism of context dependency can be characterized by **constraints** and **preferences**, entailing a need for syntactic, semantic and pragmatic[3] knowledge to resolve anaphora. In the remaining part of this section, we detail some of the different properties listed in (Carbonell & Brown, 1988) to show how complex the process of building anaphora can be (this prefigures the difficulties encountered by coreference resolution systems).

Constraints Constraints are necessary conditions that must be satisfied by the anaphora and its antecedents. They eliminate numbers of potential candidates, but also sometimes require additional knowledge or interpretation of the discourse to be discovered. For instance in:

> **[John]** and **[Mary]** went shopping. **[He]** bought a steak.

a gender constraint applies for the pronoun *he* and the only acceptable antecedent we can find is *John*, provided that we are able to derive the genders of the names. Other grammatical constraints such as number or case also apply. However, one must be careful when determining the attributes of a mention. In the following sentences:

> **[A group of students]** entered the museum. **[They]** were very loud.

the number constraint is not satisfied if we derive the number of the first NP from its syntactic head. We cannot however set the number to be plural since if we pursue the text by “a guard

[3] Pragmatics is the study of how the context built in a discourse influences its semantic.

asked the group to be more quiet", it should be considered as singular for the constraint to apply. Grammatical attributes have been used as model features in the first machine learning approaches to coreference resolution (Soon *et al.*, 2001; Ng & Cardie, 2002b).

More subtle constraints arise when dealing with selectional restrictions, requiring even more knowledge than previously. To illustrate the problem, let us take a look at the two following sentences (see Mitkov (2002) for more examples):

1. John took **[the cake]** from the table and ate **[it]**.
2. John took the cake from **[the table]** and washed **[it]**.

In such sentences, the acceptable arguments of verbs should be known to resolve the pronoun *it*. This requires an ontological knowledge with fine lexical properties, including the types of arguments a verb accepts (e.g., the argument of "eating" must be eatable). In practice, building such knowledge databases is a very hard task. But it is not the sole issue we find with constraints. If we go deeper in semantics, we can find constraints that require a dynamic interpretation of discourse:

John gave **[Tom]** an apple. **[He]** ate it.

In that sentence, a semantic constraint applies: only *Tom* can be the antecedent of the pronoun *he* since *John* does not have the apple anymore after the first sentence. Enforcing this kind of constraints in an anaphora resolution system requires a large amount of real-world knowledge and a whole reasoning process to understand what is going on, and what is permitted or not by the semantic interpretation.

Preferences At the opposite of constraints, preferences favour some potential antecedents over the others, provided that they satisfy certain properties. These are not strictly logical conditions such as constraints but factors that influence the attachment of the anaphora to previous mentions. For instance, the "case-role persistence" preference states that an anaphor tends to be employed with the same semantic case role as its antecedent:

1. Mary gave an apple to **[Susan]**. John also gave **[her]** an orange.
2. **[Mary]** gave an apple to Susan. **[She]** also gave John an orange.

In these examples, the pronoun is employed with the same role as its antecedent, object in the first example and subject in the second. This property is obviously not a strong constraint, but influences the way an antecedent is selected. These preferences are typically employed as features in machine learning based models and automatically weighted from annotated data. In the same way as for constraints, some preferences need a large amount of knowledge to be established. The two examples below:

1. Mary drove from the park to the **[club]**. Peter went **[there]** too.
2. Mary drove from the **[park]** to the club. Peter left **[there]** too.

have identical syntactic structures but, at the discourse level, there is a preference for a "pragmatic alignment" of the first and second sentences: the actions of going from one place to the other tend to be in the same direction. This example shows how subtle preferences can be and we can also imagine how difficult the task of detecting such properties is in computational models.

We just saw that coreferential anaphora resolution (and by extension coreference resolution) requires information at the lexical, syntactic, semantic and pragmatic levels. It supposes that we need to build complex discourse representations to be able to deal with all varieties of coreferential anaphora. However, even at the lowest level, a large amount of knowledge is required to derive all the linguistic properties on which the resolution depends: this can be name genders and numbers, semantic properties of object. Consequently, the performance of automated resolution systems will strongly depend on the amount and quality of pre-processed data, which make coreference resolution a very challenging task. We should always keep this in mind when designing our own coreference resolver.

2.1.3 Other kinds of anaphora

In the discussion above, we have restricted anaphora to the case when they are coreferent with their antecedent. However, the general mechanism of anaphora, which is roughly speaking the backward dependencies of such discourse element to the context, is not restricted to coreference and it can potentially involve different semantic relations. Even if we will not address these kinds of anaphora in this thesis, it is interesting to see how complex the mechanism can be. The following examples show anaphora which are not coreferential, even if the mentioned entities are semantically close to each other (Garnham, 2001; Karttunen, 1969):

1. Sally admired **[Sue's jacket]**, so she got **[one]** for Christmas.
2. The man who gave **[his paycheck]** to his wife is wiser than the man who gave **[it]** to his mistress.

In the first case, the mention "one" introduces a new entity in the discourse: it refers to a new entity which has the same characteristics as, but which is not equal to, the previous entity introduced by "Sue's jacket". In the second example, "it" does not refer to the same paycheck, but to the paycheck of another man, which is not explicitly introduced in the text. Such examples show anew the necessity of a deep understanding of the sense in the discourse to resolve anaphora.

(Clark, 1975) describes a large variety of so-called "bridging anaphora", which require more or less inference to be understood. We will not report all of them here as it goes far from our focus, but let us point out that, in many cases, there is a need of common world knowledge to resolve such anaphora (see also Asher & Lascarides (1998) for a study of bridging). In this

last example, bridging inference is required to find the anchor of an anaphora (meronymy[4]):

I visited a **[house]** last week. **[The kitchen]** was really nice.

We can see with the last example that anaphora is a very general phenomenon in discourse, which can necessitate world knowledge or common sense and semantic inference to be fully understood. We will not be going into more details about bridging anaphora, but let us mention that recently empirical methods to address bridging anaphora detection and resolution emerged (Poesio *et al.*, 2004; Lassalle & Denis, 2011; Hou *et al.*, 2013a,b), and this is now a new challenge in NLP.

2.2 Coreference resolution and related tasks

Coreference resolution is defined as the task of grouping together the mentions referring to the same entity in a text. In NLP, computational techniques designed to address this problem were similar to those for resolving coreferential anaphora. Consequently, though they are many differences from the linguistic point of view (see Section 2.1 in Chapter 1), they are almost indistinguishable from an empirical perspective, except that coreference resolution requires to find links between mentions that are not anaphora. Today's corpora have been annotated to evaluate NLP systems on coreference resolution rather than just anaphora resolution. This section specifies the vocabulary we will employ frequently throughout all the pages and defines the different tasks we have to cope with when designing an automated coreference resolver.

2.2.1 Generic vocabulary

Unlike the terminology presented in introduction (Chapter 1), the terms employed here are sometimes overloaded from a linguistic point of view to focus more on the computational aspects of coreference resolution. In fact, to be consistent with current work on coreference resolution, we favour the terminology more often encountered in computational linguistics literature over linguistics terms.

A **document** or a **text** is the framework within which we achieve coreference resolution. The task of finding coreference links across documents exists (this is called cross-document coreference resolution (Bagga & Baldwin, 1998b)), but it is almost a different problem (potentially involving mapping mentions to an ontology or knowledge database). In this thesis, we work on documents such as news articles, conversations, classical texts, texts from the web, etc.

[4]Meronymy is a semantic relation involving an entity (person, object, event) and a necessary part. For example: *tree/trunk, car/driver, race/athletes*.

Coreference should be understood broadly, though we mainly focus on resolving a restrained version of coreference. Coreference means referring to the same thing. We saw in the examples above that this can involve various mechanisms. In a text, a **mention** is a portion of a sentence, such as a noun phrase or a verbal phrase that can refer to an object of the world, or differently to an event in the case of verbs. From the point of view of a computer scientist, a mention is roughly like a pointer to a unique object of the world. Such objects are called **entities**. Two mentions pointing to the same object or event are said **coreferent**. We identify the pairwise relation of being coreferent as a set of **coreference links**.

Considering the fact that coreference is a transitive relation (if a and b are coreferent and b and c are coreferent, then a and c are coreferent), we can group together the mentions of a document being coreferent, forming **clusters** of mentions. To one cluster corresponds one entity, and because of this one-to-one mapping we will often overload the sense of this term by talking indifferently about clusters or entities.

Because we work on texts, there is a natural order on words, and consequently on mentions, and while clusters can be viewed as cliques (i.e., complete subgraphs) in the graph of coreference links, entities can also be represented by sequences or mentions. When we take the mentions of an entity in the order of the text, we refer to this sequence as a **coreference chain**. We must point out that when discussing the links of a coreference chain, we only refer to the links between two successive mentions: a cluster of n mentions has $n(n-1)/2$ links, but the corresponding chain has $n-1$ links.

When viewing entities as coreference chains, we distinguish the first mention of a chain, i.e. the mention introducing the entity for the first time. We call that mention **discourse-new**, the next mentions **discourse-old**. This denomination is easier to use: when talking about **anaphora**, one should be more careful because this implies that the understanding of the anaphora needs to go backward in the text and find an antecedent. Contrarily, two coreferring names in the text can be understood independently. A discourse-new mention will also be referred to as an **entity head**. We will also be less rigorous when naming the task of retrieving the discourse-new / discourse-old properties of a mention: we will always refer to this task as **anaphoricity detection**, as it has been traditionally called in the NLP literature.

Now we distinguish a special case of clusters, where the entity is represented only once in the text. We say that the mention (or equivalently, the cluster) in question is a **singleton**. As opposed to that, we call all remaining mentions **coreferents**.

Finally, we introduce a definition which is only related to the computational aspect of coreference resolution and does not need to be discussed in linguistic theory: we call **system mentions** or **detected mentions** the segments of text provided by an automated system which is supposed to extract mentions from raw text. Because such program is often imperfect, it does make mistakes and introduces spurious mentions in the analysis, or misses to detect true mentions. As opposed to that, **reference mentions** or **key mentions** refer to the real mentions

of the text. From an empirical perspective, these are provided by linguists on annotated corpora so as to evaluate the quality of automated coreference resolution systems.

2.2.2 Formal description of the tasks

We now formally define the task we will address in this thesis. To be perfectly clear, aside from coreference resolution, anaphoricity and singleton detection are secondary problems since they can be derived from a perfect resolution of coreference. In practice, however, they have been proven useful for improving the partitioning of mentions (we carry out a series of experiment including anaphora detection in Chapter 7, and mention and singleton detection in chapter 8). The list of tasks below is open, since any interaction with coreference could be included. For instance, relation extraction, entity typing as well as cross document coreference are problems that overlap with coreference resolution, but it is not obvious that one task is subsequent to another. In particular, extending a system to cover these tasks might require a joint approach to a more global problem. In our case, we stick to a approach to coreference resolution within a same document, which can be evaluated properly on standard data with standard metrics, and compared to existing systems.

Coreference resolution Coreference resolution is the mother task we address in our work. It is the finality, and evaluation is achieved on this task (because, in principle, we are transmitting the output to another NLP module). The coreference task we consider is defined at the document level. Formally, given a document $\mathcal{D}$ containing n mentions $m_1, \ldots, m_n$, the *coreference resolver* has to make a partition of these mentions into entities $e_1, \ldots, e_k$ such that for each entity e_l, for each pair of mentions $m_i, m_j \in e_l$, m_i and m_j are coreferent. From a practical perspective, an entity can be seen as a subset of mentions. In essence, this is a clustering task, but different approaches to the problem exist, some of them being justified by linguistic facts (see Chapter 4). Notice that, as mentioned previously, our interest is to group together coreferring mentions, and not to resolve only coreferent anaphora (the treatment of other anaphora is excluded). In Chapter 6, we address coreference resolution globally at the document level by reviewing and extending state of the art methods, compare this approach with local pairwise methods.

Anaphoricity detection As coreference resolution includes coreferent anaphora resolution (for instance with pronoun resolution), detecting anaphora is part of the task. Nonetheless, we overload the task for it to better fit in our modelling framework, and instead of considering anaphora only, we try to address discourse-new detection (hence defining a binary classification task). Formally, in the document $\mathcal{D}$, the *anaphoricity detector* has to partition the mentions $m_1, \ldots, m_n$ into two sets $\mathcal{M}_{new}$ and $\mathcal{M}_{old}$ such that, for each $m_i \in \mathcal{M}_{new}$, there is no coreference link to a previous mention $m_j \preceq m_i$ ($\preceq$ orders mentions according to their location in the text). Moreover, for each $m_i \in \mathcal{M}_{old}$, there exists $m_j \preceq m_i$ such that m_i and m_j corefers.

This task can be solved as a consequence of coreference resolution, but is traditionally seen as a prior task providing useful (but often erroneous) information to the coreference resolver. Joint approaches (i.e., both at the same time) to the problem exist, see for instance Rahman & Ng (2011); Denis & Baldridge (2007) (more details are provided in Chapter 4), and we will present our own global joint approach to the tasks in Chapter 7.

Mention detection In the definition of the coreference task, we suppose that a document comes along with a sequence of mentions. Evaluating coreference resolution given the mentions has been a subject of debate (evaluating coreference on detected mentions requires defining convenient metrics, and makes the evaluation very dependent on the preprocessing, see Section 2.5), and the "end-to-end task" would be to take a text empty of any mention annotation, and output a partition of mentions. This obviously requires to define the mentions on which the partition will be defined. This is the purpose of mention detection: given a document $\mathcal{D}$ containing a text $t_1, \ldots, t_m$ (the t_is can be seen as tokens), the *mention detector* should provide a set of sequences $m_1 = (t_{i_1^1}, \ldots, t_{i_{k_1}^1}), \ldots, m_n = (t_{i_1^n}, \ldots, t_{i_{k_n}^n})$ such that $m_1, \ldots, m_n$ are the mentions of the text. Despite its simple formulation, this problem is hard to solve exactly, and usually the output of a mention detector contains spurious mentions (like non-referential pronouns) and misses some true mentions. Moreover, the quality of the partition provided by a coreference resolver using detected mentions and evaluated against a reference set of mentions really depends on the accuracy of the detected mentions and can be dramatically impacted by them.

Singleton detection Finally, the last task we consider can again be solved as a consequence of coreference resolution. This is however a new task, introduced by Recasens *et al.* (2013), who showed that it could improve the quality of a coreference resolver. In fact, empirical evidence shows that coreference partitions contain a great proportion of singletons, and that separating them from the rest of the mentions could result in more accurate clusters. Formally, given a document $\mathcal{D}$ containing n mentions $m_1, \ldots, m_n$, the *singleton detector* separates the mentions into two categories $\mathcal{S}$ and $\mathcal{C}$ such that for all $m_i \in \mathcal{S}$, for all $m_j \neq m_i$, m_i and m_j do not corefer, and for all $m_i \in \mathcal{C}$, there exists $m_j \neq m_i$ such that m_i and m_j corefer. Detecting singletons can be seen as a pre-clustering task helping the coreference resolver, but also as a way to eliminate spurious mentions (depending on the way we train a statistical model for it, we can address both the problem of finding singletons and removing spurious mentions with one tool). In Chapter 8, we propose an improved model for detecting singletons, by exploiting a richer set of features than what is done in Recasens *et al.* (2013).

2.3 Evaluation metrics

Measuring the quality of coreference output is not a trivial issue. Even when evaluating a simple anaphora resolution system, counting the precision of the resolution of each anaphora

is not sufficient for many reasons: first, not all coreference relations are anaphoric (as we saw in 2.1.1, proper names can corefer without involving an anaphor) and second, we are not guaranteed that the output of the system is coherent, in particular, we are not ensured that it complies with the transitivity property logically required for the output.

So another way is to evaluate mention clusters (hence enforce the resolution system to output clusters), that is to compare the system mention partition with the reference partition. Very diverse systems can be compared by this means. This may involve clustering outputs which typically amounts to clustering a graph weighted by a pairwise coreference model. However some systems model coreference resolution differently so that this clustering task is avoided, for example entity-based systems (see Chapter 4 for a state of the art about these techniques).

Most coreference specific evaluation metrics define a precision and a recall. These quantities are comprised between 0 and 100% and they should be interpreted as follows: the precision is high when the clusters tend to have a small number of spurious coreference links, and the recall is high when the clustering permits to recoverer a high number of coreference links. Additionally, the F1-score, defined as the harmonic mean between precision and recall gives an idea of the balance between the two. If one looks at the surface involved by the harmonic mean, one can see that large imbalances are more penalized (which is not the case with an arithmetic average). Consequently, a “good coreference resolver" according to these metrics is not one that tends to produce a few precise links or big entities, but rather something in-between.

As we shall see below, one of the issues with clustering metrics is that they have been designed to compare partitions over the same set of mentions. It means that they do not take into account the evaluation of mention detections. As we will see, their definition does not integrate the fact that the set of reference mentions and the set of system mentions can be different. To be able to apply them in such cases, different extension have been proposed, all relying on an alignment of reference and system mentions before scoring the partitions.

2.3.1 Clustering metrics

As a clustering task, coreference resolution could be evaluated as such[5]. However, clustering metrics have not been very popular for evaluating coreference systems. Among coreference evaluation metrics, *purity* and *inverse purity* have been used in cross document coreference resolution (i.e., finding identity relationships between entity across documents. For instance, see Huang *et al.* (2009)). Consider the set of clusters output by a system (of response clusters): $\mathcal{R} = \{R_1, \ldots, R_k\}$ and the set of key clusters (i.e. correct clusters) $\mathcal{K} = \{K_1, \ldots, K_l\}$. Precision of system cluster R_i with regards to reference cluster K_j is naturally defined as the

[5]Notice that however, only very few authors addressed the task as a classical clustering task. See for example Cardie *et al.* (1999) and Yangy *et al.* (2004).

size of the intersection of clusters over the size of the system clusters:

$$Precision(R_i, K_j) = \frac{|R_i \cap K_j|}{|R_i|}$$

and purity is a weighted average of the maximal precision of each cluster:

$$Purity(\mathcal{R}, \mathcal{K}) = \sum_{i=1}^{k} \frac{|R_i|}{N} \max_j Precision(R_i, K_j)$$

where $N = \sum_{i=1}^{k} |R_i| = \sum_{j=1}^{l} |K_j|$ is the number of mentions. This measure evaluates the quality of cluster from 0 (the worst) to 1 (the best). Purity tends to penalize incorrect attachments such that sparser clusterings are favoured. Inverse purity is simply defined as:

$$InversePurity(\mathcal{R}, \mathcal{K}) = Purity(\mathcal{K}, \mathcal{R})$$

Another popular metric for comparing clusterings is the Rand Index Rand (1971), which is a pairwise metric used to quantify similarity between two clusterings, when they are defined over the same set of elements. The Rand Index is computed as follows:

$$Rand(\mathcal{R}, \mathcal{K}) = \frac{a+b}{\binom{N}{2}}$$

where a is the number of pairs that are in the same cluster for $\mathcal{R}$ and $\mathcal{K}$ and b the number of pairs whose elements are in different clusters both in $\mathcal{R}$ and $\mathcal{K}$. $\binom{N}{2}$ being the number of all possible pairs, the Rand Index is valued between 0 and 100%.

2.3.2 MUC

Vilain *et al.* (1995) introduced MUC, the first specific metric to evaluate the quality of predicted entities in coreference resolution. This metric was used in the scoring scheme of MUC6 coreference task. Basically, it computes for each true entity cluster the number of system clusters that are needed to cover it. Precision is this quantity divided by the true cluster size minus one. Recall is obtained by reversing true and predicted clusters. If we consider again our response $\mathcal{R}$ and key $\mathcal{K}$, precision and recall for MUC are defined as:

$$Precision_{MUC} = \frac{\sum_{R \in \mathcal{R}, K \in \mathcal{K}, R \cap K \neq \varnothing} |R \cap K| - 1}{\sum_{R \in \mathcal{R}} |R| - 1}$$

$$Recall_{MUC} = \frac{\sum_{R \in \mathcal{R}, K \in \mathcal{K}, R \cap K \neq \varnothing} |R \cap K| - 1}{\sum_{K \in \mathcal{K}} |K| - 1}$$

One must be very careful with the formulae above, because they suppose that clusters always contain two or more elements. It necessitates to "clean" the output of a resolution system and the key partition to be compared with by getting rid of their singletons. An equivalent way to write precision and recall for MUC is to consider the number of clusters created when intersecting a given response (resp. key) cluster with the key (resp. response) partition. If we denote this number by $p(R)$ (resp. $p'(K)$), precision and recall are defined as follows:

$$Precision_{MUC} = \frac{\sum_{R \in \mathcal{R}} |R| - |p(R)|}{\sum_{R \in \mathcal{R}} |R| - 1}$$

$$Recall_{MUC} = \frac{\sum_{K \in \mathcal{K}} |K| - |p'(K)|}{\sum_{K \in \mathcal{K}} |K| - 1}$$

MUC metric has been criticized on several aspects (Bagga & Baldwin (1998a); Luo (2005)), especially because it tends to favour systems creating large clusters. In the extreme case where one big cluster is created for all the document, the precision is not that much hampered. This is due to the fact that merging two entities will only account for recall errors.

2.3.3 B^3

B^3 Bagga & Baldwin (1998a) is a "mention centric" evaluation metric: it computes recall and precision scores for each mention, based on the intersection between the system/true clusters for that mention. Precision is the ratio of the intersection and the true cluster sizes, while recall is the ratio of the intersection to the system cluster sizes. Global recall, precision, and F1-scores are obtained by averaging over the mention scores. If $\mathcal{M}$ denotes the set of mentions, and $R(m)$ (resp. $K(m)$) the response (resp key) cluster containing mention m:

$$Precision_{B^3} = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \frac{|R(m) \cap K(m)|}{|R(m)|}$$

$$Recall_{B^3} = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \frac{|R(m) \cap K(m)|}{|K(m)|}$$

Although it does not have the balance problem of MUC to score high big clusters, this metric has been criticized for at least two reasons: first, it is distorted towards 100% in the case where there are a lot of singletons (Recasens & Hovy, 2011). Second, it somehow splits response clusters and uses them several times when aligning with key and response partitions, leading to counter-intuitive results (Luo, 2005).

2.3.4 CEAF

Luo (2005) proposed a different approach to scoring partition. The initial idea is to compute the score via a best one-to-one mapping between the response and key partitions, which is

equivalent to finding the best optimal alignment in the bipartite graph formed out of these partitions. It is more a template for defining evaluation metrics than a metric itself because it depends on the similarity function between a response and a key cluster. The most commonly used version is CEAF$_e$, with the "ϕ_4 similarity function":

$$\phi_4(R, K) = \frac{2 \times |R \cap K|}{|R| + |K|}$$

The best one-to-one mapping is computed using *Munkres assignment algorithm* (Munkres (1957)). The value is divided by the number of response (resp. key) entities for computing precision (resp. recall). The one-to-one mapping computed by CEAF can be found more intuitive, but one drawback is that all entities are weighted equally (only the size of the overlaps counts) regardless of the number of mentions they contain.

2.3.5 BLANC

BLANC (BiLateral Assessment of Noun-phrase Coreference) (Recasens & Hovy, 2011), is a variation of the Rand Index. To define this measure, we need to introduce additional quantities: let C_r (resp. C_k) the set of coreference links in the response (resp. the key), that is to say, the edges of the graph we obtain if we look at the clusters as cliques. In other words, we take all possible (non-oriented) pairs of mentions in a same cluster for all clusters of the response and of the key, respectively. We also consider the set of non-links, N_r and N_k being all possible other pairs for response and key respectively. Define:

$$P_c = \frac{|C_r \cap C_k|}{|C_r|}$$

$$R_c = \frac{|C_r \cap C_k|}{|C_k|}$$

and

$$P_n = \frac{|N_r \cap N_k|}{|N_r|}$$

$$R_n = \frac{|N_r \cap N_k|}{|N_k|}$$

which correspond to precision and recall for coreference and non-coreference links. BLANC Precision and Recall are defined as follows:

$$Precision_{BLANC} = \frac{P_c + P_n}{2}$$

$$Recall_{BLANC} = \frac{R_c + R_n}{2}$$

Something worth noticing is that the F-score defined for BLANC is not an harmonic

mean, but a simple average:

$$F_{BLANC} = \frac{Precision_{BLANC} + Recall_{BLANC}}{2}$$

BLANC is defined such that it can operate with singletons (clusters of one element). This is not possible with link-based metrics such as MUC and B^3, for which a previous manipulation consists in removing singletons before scoring. Considering the fact that singletons represent most mentions Màrquez *et al.* (2013); Recasens *et al.* (2013), it is important to have one metric that takes them into account.

2.3.6 "CoNLL score"

The official score that was used in CoNLL-2011 and 2012 Shared Task (see 2.4 below) was the average of MUC, B^3 and CEAF entity F1-scores. The choice of this score has been largely discussed, and because no consensus was obtained to choose one specific metric, the average score was chosen (Pradhan *et al.*, 2011, 2012)). This allows to evaluate different aspects of response partition through a mix of metrics. In any case, it is still useful to analyse carefully the precision/recall details of each metric, but generally a system having a good CoNLL score was at about the same rank with other metrics.

2.3.7 Metrics extensions

All the metrics presented above measure different aspects of clustering. Measuring the quality of system partition is still a current topic of research. There is no consensus on which particular metric to use, but there are many proposals of new metrics or extensions of existing metrics. We present some related work below.

Imperfect mention detection All the metrics above are defined for a *given* set of mentions to work with, and evaluation using them assumes that mention detection is perfect. This assumption is questionable as soon as one would like to evaluate a full coreference system taking raw text in input and that creates coreference clusters. Several adaptations of the metrics have been proposed: Cai & Strube (2010b) used alignments of key and response partitions to work on a common set of mentions. The alignment is asymmetrical in the sense that it is a different modification of the partitions that is applied when computing precision or recall. Pradhan *et al.* (2014) proposed a simpler alignment of partitions to extend all coreference metrics listed above, and performed a new ranking of all coreference systems which participated to CoNLL-2011 and 2012 Shared Tasks. Luo *et al.* (2014) proposed a particular extension to BLANC. In Chapter 8, we employ the alignment proposed by Pradhan *et al.* (2014), so as to

be able to compare with existing systems. An additional thing to be evaluated in the case of detected mentions is the quality of the detection itself.

Simple precision and recall give an idea of the detection quality, but do not make the difference between mentions that are harder to detect (for instance in the case of events) and easily detectable mentions. It can also involve some problems when singletons are not annotated, which is the case with CoNLL-2011 and 2012 Shared Tasks.

Different kinds of coreference links Common evaluation metrics have been criticized recently for providing score too strongly related to the technical aspect of clustering mentions and not having any linguistic perspective (Chen & Ng (2013)). Indeed, all mentions are treated the same way as an element of a cluster without distinction of any kind. Chen & Ng (2013) propose enriched metrics with grammatical typing of mentions, trying to give weight to more *informative* detected links, i.e., whose correctness should impact more an automated system. From their point of view, a name is more important than a nominal, and a nominal is more informative than a pronoun. Weighting mentions and coreference links lead to a redefinition of the classical metrics. Their experiments were achieved on reference mentions, but as mentioned by the authors, it could be extended to detected mentions.

The question of using one metric or another one, and one extension to system mentions or another is still open, though very lately efforts have been oriented towards having a public reference implementation (Pradhan *et al.*, 2014).

2.4 Corpora

From an empirical perspective, having a reliable source of data for evaluating automated coreference resolver is essential, and obtaining a high rate of agreement between annotators of coreference on a given corpus is a problem in itself. It requires a sound annotation scheme (see for instance, Müller & Strube (2006)) and a relevant measure of agreement (in the case of coreference, see Passonneau (2004)). Furthermore, with a large volume of annotated documents, training a statistical model becomes possible. Today, many corpora annotated with coreference are available in several languages. Of course, the most covered language is English, and it is also the language for which various state of the art coreference systems exist. The competitiveness existing on English puts a certain pressure on incoming systems and allows a thorough comparison of performances. The experiments we relate in this thesis have been carried out on CoNLL-2012 English corpus, which is currently the largest and the most diverse annotated corpus. After briefly describing previous English corpora, we make a detailed presentation of that CoNLL-2012 corpus.

2.4.1 MUC and ACE-2005

Previously, corpora annotated with coreference have been introduced on the occasion of evaluation campaign. The 6th Message Understanding Conference (MUC 6, muc (1995)) introduced a corpus of annotated news wires, with 318 documents, which was sufficient to consider using machine learning techniques on it. Initial machine learning based resolvers were evaluated on this corpus (see Soon *et al.* (2001); Ng & Cardie (2002b)). Singletons were not annotated in MUC.

The next reference corpus is ACE-2005 (Walker *et al.* (2006)). It contains weblogs, broadcast news, newsgroups and broadcast conversations, annotated with coreference clusters, relations, and events on three languages: English, Mandarin Chinese, and standard Arabic. One remarkable thing is that these different overlapping annotations allow to create novel experiments involving for instance the interaction between coreference and relation extraction or entity typing (for example, see Denis & Baldridge (2007); Chan & Roth (2010)). Another benefit of this corpus is that it makes it possible to test automated systems (and thus the features they are based on) on different domains. Indeed, adapting a statistical coreference resolver on each domain can result in substantial differences of performances (for experiments with a pairwise resolver, see Uryupina & Poesio (2012)). Contrarily to MUC, ACE contains annotations of singletons.

2.4.2 CoNLL-2011 and 2012 Shared Tasks

CoNLL-2011 and 2012 Shared Tasks, respectively entitled "Modelling Unrestricted Coreference in OntoNotes" and "Modelling Multilingual Unrestricted Coreference in OntoNotes" (Pradhan *et al.*, 2011, 2012), came with a new annotated corpus. Based on the OntoNotes v5.0 corpus (Hovy *et al.*, 2006) (English, Chinese, Arabic), it contains pre-processed information such as POS tagging, syntactic trees, lemmas, word senses, predicate-argument structures and 18 types of named entities along with coreference chains annotations. The English and Chinese language parts comprise news wire and magazine articles, broadcast news, broadcast conversations, web data, and conversational speech for an amount of about one million words per language. The Arabic part contains about 300k of news wire articles. Two versions of preprocessed information are available: *gold* and *auto*. The *gold* version contains high-quality hand-annotations, whereas *auto* information was produced by automatic state-of-the-art tools. In this thesis, we only carry out our experiments on the English part on the 2012 corpus, leaving the rest for future work. Like MUC, and contrary to ACE, CoNLL Shared Task corpus does not contain singleton annotation.

What is annotated in OntoNotes? A large range of entities and events are covered by the coreference annotation scheme. To give an idea of what is marked and what is not in the corpus, we briefly detail the annotation scheme presented in Pradhan *et al.* (2011, 2012). This

is highly important to know what kinds of phenomena are annotated, and in which quantity, in order to design a system that performs well:

- **Noun phrases:** NPs concern most of the annotated mentions. In the annotation process, all NPs are automatically extracted from gold parse trees and manually clustered by annotators, who also add supplementary mentions (e.g. verbs, possessive pronouns).

- **Verbs:** verbs are added as single-word mentions. They can corefer with other verbs (e.g. dissolve / dismantle), pronouns, morphologically similar nominalizations (e.g. discuss / discussion) and noun phrases referring to the same event (e.g. discuss / the conversation).

- **Pronouns:** all pronouns and demonstratives, except pleonastic pronouns and generic *you*, can be part of coreference clusters. In the next example, starred brackets indicate pleonastic pronouns and generic *you*, which are not annotated in the corpus:

  > Senate majority leader Bill Frist likes to tell a story from his days as a pioneering heart surgeon back in Tennessee. A lot of times, Frist recalls, **[you]*** 'd have a critical patient lying there waiting for a new heart, and **[you]*** 'd want to cut, but **[you]*** couldn't start unless **[you]*** knew that the replacement heart would make **[it]*** to the operating room.

- **Generic mentions:** generic nominal mentions (e.g. bare plurals) can be linked with corefering pronouns, but not with other generic mentions.

  > **[Parents]$_{\mathbf{X}}$** should be involved with **[their]$_{\mathbf{X}}$** children's education at home, not in school. **[They]$_{\mathbf{X}}$** should see to it that **[their]$_{\mathbf{X}}$** kids don't play truant; **[they]$_{\mathbf{X}}$** should make certain that the children spend enough time doing homework; **[they]$_{\mathbf{X}}$** should scrutinize the report card. **[Parents]$_{\mathbf{Y}}$** are too likely to blame schools for the educational limitations of **[their]$_{\mathbf{Y}}$** children. If **[parents]$_{\mathbf{Z}}$** are dissatisfied with a school, **[they]$_{\mathbf{Z}}$** should have the option of switching to another.

- **Pre-modifiers:** proper nouns in adjectival form cannot be marked as coreferent (e.g. adjectival forms of geopolitical entities). Dates and monetary amounts, when used as modifier, can also belong to a coreference cluster.

  > During World War II, the two hundred thousand strong **[Chinese]*** Expeditionary Force went abroad to fight. The Burma Road was the lifeline artery for **[China 's]$_{\mathbf{X}}$** anti-**[Japanese]*** battlefields. The Tengchong Battle, an annihilation of the **[Japanese]*** army , became a typical example for the **[China]$_{\mathbf{X}}$** war area.

> The company's **[$150]**$_{Y}$ offer was unexpected. The firm balked at **[the price]**$_{Y}$.

- **Copular verbs:** attributes in copular structures[6] are not linked to the referent they modify and are thus not considered as mentions:

> But **[Wal-Mart]**$_{X}$ is really **[the king of the bottom line]***, aren't they?

- **Small clauses:** these constructions having the same characteristics as copulas (subject-predicate) are likewise not annotated:

> John considers **[Fred]*** **[an idiot]***.

- **Temporal expressions:** deixes such as "now" or "yesterday" can be linked to other temporal expressions. This may involve complex constructions such as in the following example:

> The limit could range **[from three years to seven years]**$_{X}$, depending on the composition of the management team and the nature of its strategic plan. At **[the end of [this period]**$_{X}$**]**$_{Y}$, the poison pill would be eliminated automatically, unless a new poison pill were approved by the then-current shareholders, who would have an opportunity to evaluate the corporation's strategy and management team at **[that time]**$_{Y}$.

- **Appositives:** a specific annotation schema is used for these constructions, however, they are ignored in evaluation. Thus in data, only a bigger mention covering the appositive is marked:

> "The risks for sterling of a bad trade figure are very heavily on the down side," said **[[Chris Dillow]*, senior U.K. economist at Nomura Research Institute]**$_{X}$. [...] However, **[Mr. Dillow]**$_{X}$ said [...].

We give some statistics about the type of annotated mentions in Table 2.1: overall we have about 45% of pronouns, one quarter of commons nouns, one quarter of proper nouns, and a small portion of verbs. However, if we look at the details per category, we see that the distribution is variable, and really depends on the type of document. For instance in TC (phone conversations) most mentions are pronouns while in NW (news wire) there are fewer pronouns than common or proper nouns. This suggests that machine learning approaches to coreference resolution should take into account the document category (e.g. one model per category). In practice, using a statistical models on new categories would require additional techniques such as domain adaptation (which is not the topic of this thesis).

[6] Some common copular verbs are: *appear, be, become, feel, get, go, keep, grow, lie, look, prove, remain, resemble, run, seem, smell, sound, stay, taste, turn*.

	BC	BN	MZ	NW	PT	TC	WB	total
pronouns	10,039 (53.9%)	7,544 (33.8%)	4,977 (36.9%)	8,044 (23.3%)	23,434 (55.6%)	9,418 (78.0%)	5,939 (48.1%)	69,395 (44.6%)
common nouns	4,247 (22.8%)	6,996 (31.3%)	3,577 (26.5%)	12,856 (37.2%)	9,940 (23.6%)	1,492 (12.4%)	3,305 (26.8%)	42,413 (27.3%)
proper nouns	3,889 (20.9%)	7,249 (32.5%)	4,773 (35.4%)	12,993 (37.6%)	8,383 (19.9%)	962 (8.0%)	2,932 (23.7%)	41,181 (26.5%)
verbs	435 (2.3%)	541 (2.4%)	150 (1.1%)	620 (1.8%)	380 (0.9%)	207 (1.7%)	172 (1.4%)	2,505 (1.6%)

Table 2.1: Distribution of mentions (train)

The coreference task on *OntoNotes* The task on *OntoNotes* is twofold: the first stage consists in detecting mentions (whether they are anaphoric or not is not important), and the second stage is coreference resolution. It is crucial to notice that the output of coreference resolution is supposed to be free of singletons, that is to say, mentions being the only ones referring to an entity are removed from the system partition (indeed, singletons are not annotated in the gold). This affects both the mention detection metric and some of the coreference metrics, but also suggests that the task of detecting singletons might have a significant impact on the coreference system performance. In fact, Recasens *et al.* (2013) showed that incorporating singleton detection as a preliminary task to coreference resolution improved the performances of a state of the art coreference resolver (Lee *et al.*, 2011).[7].

There are three levels of difficulty in *CoNLL-2011* and *2012* tasks, according to the amount of predicted information we start with: the official complete task is "predict mention and resolve coreference", the first supplementary task is "given gold mention boundaries, resolve coreference" and the second is "given gold mentions, resolve coreference". The difference between the two supplementary tasks lies in the fact that "gold mentions" do not include singletons whereas "gold mention boundaries" do. However, a quick examination of the data shows that gold mention boundaries contain also non-referentials, i.e. non mentions, so that it might not be the best setting for evaluating the clustering part in coreference resolution.

Systems that only use information contained in the CoNLL files (for English, supplementary gender data from Bergsma & Lin (2006) is allowed) are evaluated in the *closed track*. If a system uses additional information such as data extracted from *Wikipedia*, it is evaluated in the *open track*. Most of the systems that took part into the shared task had no *open track* version and relied only on provided information.

The system participating to the tasks were ranked using CoNLL score, i.e. the unweighted average of *MUC*, B^3 and *CEAF* (entity version)[8].

[7] We present some improvements of their singleton model in Chapter 8

[8] The winner were *Standford* rule based system in 2011 (Lee *et al.*, 2011) and a structured perceptron based system in 2012 (Fernandes *et al.*, 2012))

	BC	BN	MZ	NW	PT	TC	WB	total
verb mentions	435	541	150	620	380	207	172	2,505
all mentions	18,553	22,254	13,421	34,397	41,919	12,047	12,287	154,878
ratio	2.3%	2.4%	1.1%	1.8%	0.9%	1.7%	1.4%	1.6%

Table 2.2: Amount of verb mentions (train)

2.4.3 Verbs clusters in CoNLL-2011/12

CoNLL-2011 and -2012 "unrestricted coreference" annotations do not only cover noun phrases and pronouns but also verbs, which can corefer with other verbs or with noun phrases. Because the amount of verbs within annotated mentions is very small (about 1.6% of the training set), this aspect has widely been neglected by the coreference systems participating to the tasks: in 2011 only three systems out of eighteen took verbs into account (Xiong *et al.* (2011); Yang *et al.* (2011); Zhekova & Kübler (2011)) and in 2012 Li (2012) was the only one (for sixteen participants). Moreover, those systems were ten points behind the best performing systems. The usual method for detecting mentions consists in extracting all noun phrases, pronouns, and named entities and perhaps eliminating some of them (e.g amounts, pleonastic "it", some dominated NP, etc) using predefined patterns. All extracted elements are then cast into the clustering process.

Ignoring verbs does not really affect link-based evaluation metrics such as MUC and B^3, but there is a chance that the outputs will be penalized by entity-based metrics (CEAFe) if verbs form a larger proportion of entities. In our presentation of the corpus, it is interesting to investigate the data and describe the behaviour of verb clusters and show that there is indeed a consistent pattern that should not be ignored when designing unrestricted coreference resolution systems.

Proportion of verbs in CoNLL-2012 (English part) The statistics of this part are computed using counts on the training set of CoNLL-2012 English. Verb mentions are captured as mentions with a single token tagged as a verb (i.e. VB, VBD, VBG, VBN, VBP or VBZ). Since we use gold mentions, singletons are ignored. If we examine the amount of verb mentions, by category (table 2.2), we see that, even if small variations appears, they are marginal compared to all mentions.

We now compute, by category, the proportion of verb entities defined as entities containing at least one verb mention (table 2.3). With larger variations, we observe that verb entities are not trifling any more (these form one entity out of ten in BC and TC). We also see that the number of verb entities is closed to the number of verb mentions, which suggests that verb entities are small.

	BC	BN	MZ	NW	PT	TC	WB	total
verb entities	410	504	146	585	375	204	169	2,393
all entities	4,233	6,431	3,532	9,403	6,606	1,913	2,991	35,109
ratio	9.7%	7.8%	4.1%	6.2%	5.7%	10.7%	5.7%	6.8%

Table 2.3: Amount of verb entities

metric	precision	recall	F1
MUC	100.0	97.92	98.95
B^3	100.0	98.27	99.13
CEAF	98.39	98.39	98.39
CEAFe	91.39	97.88	94.53
BLANC	99.86	99.48	99.67
CoNLL score			97.53

Table 2.4: Perfect resolver without verb mentions

Impact on metrics Before going any further, let us estimate the impact of detecting verb entities in unrestricted coreference resolution. We set up two scenarios to approximate the loss due to ignoring verbs: in both cases we use a perfect resolver (i.e. find all coreference links between the detected mentions), in the first case we remove all verb mentions and in the second case all verb entities. When removing verb mentions only, the oracle can find links between the remaining parts of verb entities, which is a little bit too optimistic. On the other hand, removing the entire verb entities amounts to resolving these parts as singletons.

Patterns for verb entities We plot the distribution of verb entity size by category (see figure 2.1), removing sizes exceeding 8 which concern less than 0.7% of verb entities. As we can see, there are minor variations among the different document categories but overall verb entities are very small: in fact 93% have size 2 or 3. Given that at least one mention is a verb, we can seek a general behaviour of the other part of the entity.

Searching for coreference links involving verbs, we can find some that seem pretty hard to solve because they suppose a deep understanding of the underlying semantic :

Then yesterday he **resigned** [...]

metric	precision	recall	F1
MUC	100.0	97.32	98.64
B^3	100.0	97.92	98.95
CEAF	97.92	97.92	97.92
CEAFe	89.27	97.44	93.17
BLANC	99.85	99.32	99.58
CoNLL score			96.92

Table 2.5: Perfect resolver without verb entities

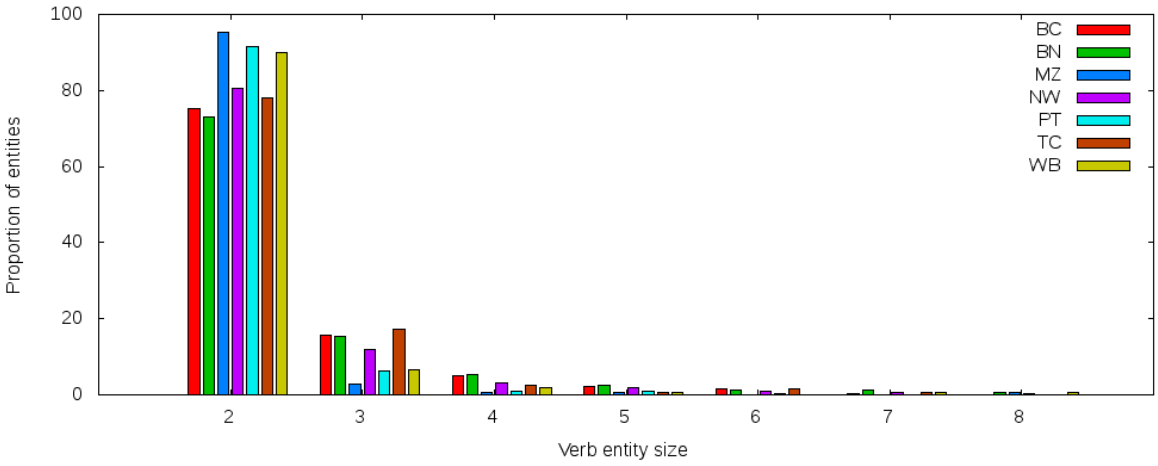

Figure 2.1: Verb entity size distribution

Barak didn't just do **some one thing which is very cynical**, he also did **something which is not very clever politically** [...]

Hopefully, this kind of coreference links is not the majority, and we can find more local patterns which are more likely to be learned. As a short conclusion, we saw that verb mentions were not addressed by the majority of state of the art coreference resolvers, and that this was not really penalized, considering the fact that removing them has a rather limited impact. Indeed, if one has the only objective to maximize the evaluation metrics with his/her resolver, ignoring verbs can be a good strategy because it permits to avoid introducing more noise during mention detection. In our experiments, we have specific features for verbs when evaluating on gold mentions, but we do not try to detect them.

However, it would be interesting to give them more importance as they provide a more challenging task from the NLP point of view. Using an incentive for addressing verb coreference would be to employ weighted metrics such as those proposed by Chen & Ng (2013), with a setting that gives a better reward to coreference links including verbs. Concerning the experiments we carry out in this thesis, we address the problem of solving coreference links (when working on gold mentions), but not the problem of detecting verb mentions, which is postponed to our future work.

2.5 Gold mentions vs detected mentions

We conclude this chapter by a discussion which, from our point of view, has a great importance for evaluation. We saw above that the CoNLL-2012 Corpus, the most complete corpus, on which we will carry out our experiments, does not annotate complete gold partitions (i.e., including singletons), but only provides mentions that have at least one coreferent counterpart. Additionally, gold mention boundaries do not correspond to the mentions, they are a superset of them (i.e., they contain spurious mentions).

Now, when we evaluate a coreference resolver, we can distinguish two aspects, corresponding to two processing steps: mention detection, and mention partitioning. However, the output is evaluated against the gold mentions of the corpus, that is, the partition short of its singletons, with both coreference metrics and the mention detection metric (a simple precision/recall metrics on gold mentions). Thus, both detection and partitioning are evaluated at the same time, and it is difficult to say if an improvement of the whole system comes from an improvement of the mention detector or from the clustering method. For example, Kummerfeld *et al.* (2011) show a significant improvement due to the sole addition of filters to the mention detector, suggesting that evaluation of coreference on detected mentions is very sensitive to the detection quality.

The methodology we chose for properly separating mention detection and clustering is to evaluate our clustering models on gold mentions only (i.e., gold mentions are given, without singletons, and need to be partitioned), to highlight the improvements we get on this part (see Chapters 5, 6 and 7), and make a final evaluation of the complete system in a separate work (Chapter 8). In this way, we can set the focus on the clustering part first, and then add the dependency on mention detection in a second stage. Of course, we are conscious that gold mentions is an easier task than the complete track, and our results on gold mentions should only be compared to the results obtained by other systems in the same track.

It is important to emphasize the fact that evaluation on gold mentions has been stable (because metrics are well defined), whilst several issues happened on detected mentions, which led to re-evaluation of all systems using new implementations of the metrics (Pradhan *et al.*, 2014).

2.6 Chapter summary

This chapter first introduced the vocabulary we use throughout this thesis (see Section 2.2.1). We next make a detailed introduction to the tasks we will address in our experiments (Section 2.2.2), namely, anaphoricity detection, mention detection, singleton detection and coreference resolution itself. These coreference-related tasks can be, in principle, resolved through the sole treatment of coreference resolution, and some of them are not addressed directly by state of the art systems. After that, we describe the different evaluation metrics employed in coreference resolution (see Section 2.3). It is worth noticing that finding relevant and sound evaluation metrics is still an active topic. We present the main corpora used to evaluate coreference resolvers in English in Section 2.4, providing more details on CoNLL-2012 corpus, on which we carry out all our experiments. Finally, we add a discussion on the evaluation methodology (see Section 2.5). We state that, on CoNLL-2012 corpus, evaluation on *gold mentions* is a good way to see improvements in the partitioning procedure, while the evaluation on detected mentions introduces a dependency on the quality of mention detection.

Chapter 3

Preliminaries in Machine Learning

In this chapter we review a few aspects of machine learning that we will often refer to in the next chapters. This will give us the opportunity to detail the theoretical properties of the models we employ in our work. We start by drawing up a brief outline of Machine Learning *(section (3.1)). In Section 3.2, we formally describe the problem of representing NLP objects by features, and review some issues related to choosing features. Then, we move on to introducing linear models and detailing perceptron-like learning algorithms, some of them with a large margin property (Section 3.3). We make a brief review of kernel methods in Section 3.5. We terminate by explaining how the previous learning algorithms can be adapted to structure learning in Section 3.6.*

3.1 Introduction

Machine Learning (ML) is a discipline that emerged from computer science and statistics in the mid-twentieth century, and grew rapidly in importance to become central in many applications such as pattern recognition, data mining, games, vision, robotics and of course language processing. Its growth is partly due to the fact that more and more data is available in diverse areas (for instance, annotated corpora allow to design ML-based systems for coreference resolution). Basically, ML techniques are used to automatically discover regularities in data, which can appear tedious for anybody who wants to explore a large amount of data, and use them to make decisions such as classifying new data points. It is indeed the ability of ML methods to make use of both algorithms and statistics, and therefore avoid the process of manually designing rules, exceptions to the rules, exceptions to the exceptions and so on, that makes it attractive in many fields in which data are available. To keep it simple, we can distinguish two problems in ML:

- **Supervised learning** techniques are employed to infer functions. The goal is to learn a mapping $\mathcal{X} \to \mathcal{Y}$ from labeled data (also named *training set*), that is, pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. The mapping will be used to label new data objects from $\mathcal{X}$. There are many ways to tackle this problem depending on, for instance, how data is represented (e.g. directly in a vector space or with parametric probability distributions), what learning algorithm is used, the objective selected for optimization or the complexity of the model (i.e. the

number of parameters to tune; one has to find a model with a good balance between the variance of its output and a low bias, that is to say, enough flexibility to fit the data well).

- **Unsupervised learning** techniques are used when the objective is to discover patterns in the data. In this case, the model is provided only with unlabeled data from a set $\mathcal{X}$, and the goal can be (among others) to discover similarities in the data and form groups or *clusters* of proximate objects, or to determine the distribution of data (also known as *density estimation*).

However, the frontier between these two approaches is clearly define since one can introduce more or less supervision to unsupervised techniques, allowing to set up models that adapt easily on different domains. Examples can be found in coreference resolution in Poon & Domingos (2008), where a little supervision for finding coreference links is introduced through appositions and predicate nominatives.

ML techniques have been widely applied to a variety of NLP tasks: POS tagging, parsing, named entities recognition, relation extraction between entities, machine translation, coreference resolution, etc. Unlike rule-based systems, they have shown robustness when encountering previously unseen data or noisy inputs. As a consequence, tackling a NLP task with ML techniques partly amounts to defining a good representation of the items to be input in the learning models. This is the object of the next section. For the rest of this chapter, we are only concerned in supervised learning and especially in *classification* (i.e. learning a mapping to discrete categories).

3.2 Modeling the problem and selecting features

This section is dedicated to underlying the importance of choosing a representation for the instances of a problem. Choosing a representation is related to what is done during *data preprocessing*. In certain cases, data preprocessing turns out to be major part of a model with regards to the size of source code concerning this part, which happens for instance in coreference resolution system using machine learning.

3.2.1 Feature representation of the data

Once we have decided to rely on machine learning techniques to address our NLP tasks (primarily coreference), we have to design a formal representation of the problem, which is associated to a collection of **features** through which the computer manipulates the inputs. This is a full part of the modeling process, and some crucial decisions are taken at this stage. Formally, choosing such representation amounts to designing a mapping Φ that transforms raw objects

(set $\mathcal{X}$) into observable items (set $\mathcal{F}$) :

$$\Phi : \mathcal{X} \rightarrow \mathcal{F}$$

In this thesis, we restrict ourselves to working with $\mathcal{F}$ as a feature space[1], which is basically a vector space $\mathbb{R}^d$. Also depending on the learning algorithms used downstream, the choice of Φ strongly impacts the quality of a system based on ML. A bad representation of data may involve low scores in evaluations.

Before going into feature selection, we detail the process of generating features. In fact, the first step is to give the problem we tackle a representation that can be handled by learning models. A set of initial features is defined manually or by the help of templates. They can be extracted from a physical signal like in image understanding and speech recognition, or for our NLP tasks, from text and pre-processing units (parser, tagger, etc). Features can be :

- Real numbers (e.g. the average number of NP per sentence).
- Integers (e.g. number of words in a NP).
- Booleans indicating whether the object satisfies a given property or if it has a certain attribute (e.g. if a NP is plural).
- Categorical (having values in a finite set). If we need a full real representation of features, that is to say if we want $\mathcal{F}$ to be a vector space[2], we use binary representations of categorical features: for categories $\{A, B, C\}$, we have 3 Booleans indicating which is the value of the categorical feature (A is $(1, 0, 0)$, B is $(0, 1, 0)$ and C is $(0, 0, 1)$).

When processing text, the amount of features can be considerable if, for example, we use lexicalized features for which there can be as many features as words appearing in certain patterns in the data. Among these initial features, some are nonetheless not relevant or not informative for the task we are trying to solve and they should therefore be eliminated from the representation of data. So an additional work on features is needed: this is *feature selection*. Feature selection is also called *variable subset selection*, which is more explicit on the fact that we are facing an exponential number of possible mappings Φ (see 3.2.2 for more details). To sum up, there are two preliminary steps to design a system using machine learning: feature generation and feature selection[3]. In a supervised learning setting, we start from a collection of labeled raw objects $\{(o_t, y_t)\}_{0 \leq t < N}$, and transform them into the feature representations $\{(\Phi(o_t), y_t)\}_{0 \leq t < N}$ on which the learning model will work.

[1] One could avoid to use specific features, and work directly on the similarity of objects using kernels (Bakir *et al.*, 2007). However the above framework encompasses all the methods we employ in this thesis.

[2] This happens for instance when we work with linear models.

[3] But there are special cases when feature selection is achieved during leaning, for example when using L1-regularization.

3.2.2 Feature selection

There are numerous strategies for feature selection, we present a few that are used in NLP (we will introduce our own method for selecting features for coreference resolution in Chapter 5):

- The simplest approach is to measure the relevance of each feature separately, rank them, and take the n *best* (so n is a parameter of the method, and it controls the tradeoff between the accuracy of features and the complexity of the model). Two popular ways to evaluate relevance in a supervised classification framework is to compute *correlation coefficient* or *mutual information* between the features and the class labels (seen as random variables – for instance, see Peng *et al.* (2005)).

- Instead of taking the n-best features, another approach consists in achieving *greedy forward selection* or *greedy backward elimination*. The first method adds the feature leading to the best improvement of the system and stops when it cannot find any improvement. The second does the same but the other way around, by removing features until no improvement is possible. These strategies may be useful when the output of the learning model is transformed and evaluated with specific metrics, which is the case of coreference resolution (see Chen & Ng (2012) for a system using a greedy backward feature selection).

- In the presence of many irrelevant features, we have to prevent the model from *overfitting* the training data, which may results in bad predictions on unseen data (if the model fits the training data too much, there is risk that it may be unable to make good predictions on unseen examples). Regularization methods encourage the fitted parameters of a model to be small. In particular, L1-regularization encourages to set many parameters to zero during learning, which is equivalent to removing features from the model. This produces sparser models, and this is not only interesting when we have many more features than training samples, but also when a great part of those features are irrelevant. L1-regularization has shown to reduce significantly the misclassification error in that last configuration (Ng, 2004a).

Considering that the set of features is fixed, we have to choose what kind of learning model we will use. We must keep in mind that many constraints should be taken into account: the size of the training set, the number of features and the computational complexity of the learning algorithm are decisive in the choice of the model. In the next sections, we detail the models and learning algorithms we used in this thesis.

3.3 Linear models for classification

Linear models have been widely used in statistics, where they are often associated with regression. Here we are interested in linear models for classification. In that case and in all of this section, $\mathcal{F}$ is a real vector space $\mathbb{R}^d$ (categorical features are binarized) and the class of a feature vector depends on linear combinations of its components. This actually means that predictions do not take feature interactions into account: the weight associated to one feature is fixed regardless of the values of the other features. However linear models have a significant advantage in that they are associated with fast learning algorithms able to handle a large amount of training examples with very large feature spaces and requiring small memory. These are called *online learning algorithms*. Moreover, using *kernels* (see Section 3.5), it is still possible to use the framework of linear prediction and apply its features to highly non-linear predictors. Let us now formally introduce those models.

3.3.1 Binary classification

Let us start by considering binary classification (a more detailed introduction can be found in Bishop (2007, chapter 4)). When dealing with linear models it is more convenient to use the following labels: $\mathcal{Y} = \{-1, +1\}$. Examples of the training set associated to label -1 are called *negative instances*, and those marked by label $+1$ are *positive instances*. A linear model separates positive instances from negative instances in $\mathbb{R}^d$ with a hyperplane parameterized by the equation $w \cdot x + b = 0$. More precisely, for $x = (x_1, \ldots, x_d) \in \mathcal{F}$, the class prediction $\hat{y}(x)$ for x is given by:

$$\hat{y}(x) = \text{sign}(w \cdot x + b) \qquad \text{where sign}(z) = \begin{cases} -1 & \text{if } z < 0 \\ +1 & \text{if } z \geq 0 \end{cases}$$

where $w = (w_1, \ldots, w_d)$ is a *weight vector* and b is the *bias* (it gives the shift of the hyperplane oriented by w to the origin). These are the parameters of the model. We can include the bias into the weight vector by taking $x = (1, x_1, \ldots, x_d)$ and $w = (b, w_1, \ldots, w_d)$ we get a simpler expression for prediction :

$$\hat{y}(x) = \text{sign}(w \cdot x)$$

The real non negative number $|w \cdot x|$ can be seen as a *score* of confidence in $\hat{y}(x)$: the higher it is, the greater is the confidence in this prediction. Before going directly into detailing how the weights are set from the training set and a learning algorithm, we shall extend the linear prediction to multiple classes.

3.3.2 Multiclass

The most direct way to achieve multiclass predictions with linear models is to define one weight vector per category and take the best scoring category as final prediction. Suppose we have K classes and weight vectors $(w_i)_{1 \leq i \leq K}$, the predicted class for a feature vector x is given by:

$$\hat{y}(x) = \underset{1 \leq i \leq K}{\text{argmax}}\ w_i \cdot x$$

Learning can simply be achieved separately for each class by using the instances of the class as positive examples and all other instances as negative examples. So in this case, multiclass prediction reduces to some binary classifications. However, as we will see in the next sub-section, learning multiclass models can be a little tricky if we want to take into account the multiclass structure during learning.

3.3.3 Structures

Another framework, more general than multiclass, is structured prediction: the purpose is to learn a mapping $\mathcal{X} \rightarrow \mathcal{Y}$ to structured outputs, which in practice may be sequences, trees or graphs. This is very useful in domains such as NLP since many tasks amount to computing structures that have internal dependences (for example a sequence of POS tags, a parse tree). Because the combinatoric is much bigger than for multiclass problems, the architecture of the models is different. Collins (2002) introduced perceptron-based linear models to handle such structures: the product $\mathcal{X} \times \mathcal{Y}$ (instead of $\mathcal{X}$) is mapped to a feature space:

$$\Phi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$$

Φ is called a joint feature map. Then each pair of object and structure can be scored using a linear model, and taking the best match is straightforward:

$$\hat{y}(x) = \underset{y}{\text{argmax}}\ (w \cdot \Phi(x, y))$$

However, computing the argmax cannot be done naively by enumerating all the possible objects y since the combinatoric is tremendous (often exponential or more in the size of the object): instead, it is more clever to insert a search algorithm in the computation such as Viterbi algorithm for sequence calculation or Prim algorithm for mapping to a tree. Nevertheless, to make these algorithms compatible with the computation of the argmax, the choice of Φ is restricted. For example, mapping an object to a tree can be achieved by taking $\Phi(x, y)$ as a sum of feature vectors associated to the edges of the tree [4], so that Prim algorithm[5] can work

[4] We will use these kinds of representations in Chapters 6 and 7.

[5] Prim algorithm is used to compute minimum or maximum weighted spanning tree on an undirected graph.

on a weighted graph. We will see in details how to handle the structured learning procedure with linear models in Section 3.6.

3.4 Online learning algorithms

So far we described how to make predictions with linear models, but we did not explain how the weight vectors were set up. An important distinction has to be made here. There are two ways for designing learning algorithms: the first allows to load all the training set in memory and works on all the feature vectors at the same time. This is called *batch learning*. The other approach consists in successively loading one example of the training set and using it to update the weights. That is *online learning*. Batch learning methods are often theoretically well founded and provide good results, but in reality it is difficult to cope with large datasets since they would require a tremendous amount of memory and computation time. Online learners only see the data sequentially, and may require several passes over the training set, but their running time is more reasonable and, since a lot of work has been done lately to reduce the performance gap with batch learners, they offer acceptable performances. Here we are particularly interested in *online algorithms*: they are simple to implement and well suited to running many experiments in a moderate time so as to compare different settings rapidly. Moreover, linear methods perform well on the problems discussed in this thesis, provided we have a good selection of features (see Chapter 5). In this section we do not deal with feature representation, we still suppose that $\mathcal{F} = \mathbb{R}^d$ and we write x_t instead of $\Phi(o_t)$. In addition, we only give the full description of learning procedure for binary classification.

3.4.1 Perceptron algorithm

The Perceptron was introduced by Rosenblatt (1958) as a model of how information are processed and stored by the brain. Rapidly, it was proven that if a training set is linearly separable, that is if there exists a hyperplane separating the instances of each classes in $\mathcal{F}$, then the perceptron algorithm is guaranteed to fit the parameters to find a separation in a finite number of steps. Later, the perceptron was criticized for its inability to handle non linearly separable problems and the model was considered to be doomed to disappear. However, this algorithm is still used nowadays, especially the "averaged perceptron" which is very popular in NLP (especially its structured version), for example in POS tagging (Collins, 2002), parsing (Collins & Roark, 2004) or coreference resolution (Bengtson & Roth, 2008; Stoyanov *et al.*, 2010a).

Concretely, the Perceptron algorithm for fitting a linear model proceeds in rounds. During each round, the algorithm goes through all training examples sequentially, tries to classify each one using its current weights and updates the weights whenever the classification is incorrect. The complete procedure for binary classification is given in Algorithm 3.1. The "receive" instruction should be understood as follows: if $\mathcal{D} = (x_t, y_t)_{0 \leq t < N}$, it is actually $x_{(t \bmod N)}$ that

INPUT: Training data: $\mathcal{T} = \{(\boldsymbol{x}_i, P_i\}_{i=1}^{T}$, Iterations: N, Learning rate: τ
OUTPUT: Weight vectors $\boldsymbol{w}$

```
Initialize: w_1 = (0, ..., 0)
for n : 1..N do
    for t : 1..T do
        Receive instance: x_t
        Predict: ŷ_t = sign(w_t · x_t)
        Receive correct label: y_t ∈ {−1, +1}
        if ŷ_t ≠ y_t then
            Update: w_{t+1} = w_t + τ y_t x_t
        else
            Proceed: w_{t+1} = w_t
        end if
    end for
end for
```

Figure 3.1: Perceptron learning algorithm for binary classification.

is received since the procedure cycles on the training examples. Two criteria can be used to stop iterating: a limit on the number of cycles on the training data or stopping when the loss on the training data does not vary too much between two consecutive iterations. The loss is a function that quantifies the error of the model on a prediction. Let us pick up an example (x_t, y_t) and consider the current weight vector w_t. Common losses used in classification problems are[6]:

- *0-1 loss* : $\mathbb{I}_{(y_t \neq w_t \cdot x_t)}$ (a binary indicator of the error)
- *hinge* (or *max-margin*) *loss*[7]: $\max\{0, 1 - y_t(w_t \cdot x_t)\}$

Now let us discuss some properties of the Perceptron algorithm. First, to have an intuition of how it works, we can see the effect of an update with an instance (x_t, y_t) on a further classification of x_t: when x_t is misclassified, $w_{t+1} = w_t + \tau y_t x_t$ and then $w_{t+1} \cdot x_t = w_t \cdot x_t + \tau y_t \|x_t\|^2$ so that $\text{sign}(w_{t+1} \cdot x_t)$ has more chances to be y_t. In fact, if the *learning rate* τ is large enough, the effect is immediate and x_t is well classified after the update, but overall the hyperplane moves too much at each update and this behavior may hamper the global performance of the perceptron, especially if the training data is noisy. On the other hand, if τ is too small, the hyperplane moves too slowly and a lot of iterations over the training set are required to classify the instances correctly. The two following theorems guarantee convergence of the perceptron in the case of *linear separability* and give a bound of the number of errors made by the learning algorithm.

Theorem (Rosenblatt, 1958) When the training data is *linearly separable*, i.e. when there exists a hyperplane with parameter ζ such that: $\text{sign}(\zeta \cdot x_t) = y_t$ for all t, then the perceptron

[6] Task-specific losses in structured predictions are described in Section 3.6.
[7] The hinge loss has the strong advantage to be convex, which will be used by passive-aggressive algorithms (see 3.4.2).

will find a separating hyperplane in a finite number of iterations, i.e. there is a number T such that: $\text{sign}(w_T \cdot x_t) = y_t$ for all t.

Theorem (Novikoff, 1962) Take $\tau = 1$ and $\mathcal{D} = (x_t, y_t)_{0 \leq t < N}$. Let us take $R > 0$ such that $\|x_t\| \leq R$ for all t. Suppose that there exists $u \in \mathbb{R}^n$ and $\gamma > 0$ such that $\|u\| = 1$ and $y_t(u \cdot x_i) \geq \gamma$ for all t. Then the number of mistakes made by the perceptron algorithm in one round over $\mathcal{D}$ is at most $(R/\gamma)^2$.

The latter theorem exhibits the notion of *margin*, which geometrically corresponds to the largest stripe around a separating hyperplane which does not contain any data points. Formally, if u is a unit vector parameterizing a hyperplane separating $\mathcal{D} = (x_t, y_t)_{0 \leq t < N}$, the geometrical margin is the distance of the hyperplane to the closest point of $\mathcal{D}$: $\min_{0 \leq t < T} \frac{|u \cdot x_t|}{\|u\|}$.

Another property of the perceptron algorithm is that it only needs to store the current weight vector w_t and nothing else, and it reads the training set sequentially during learning: it is in the category of online learning algorithms.

Finally, an important modification of the perceptron is the averaged perceptron (see Freund & Schapire (1999)), which has shown to provide good results on NLP tasks in practice (for example: Fernandes *et al.* (2012)). It consists in learning with the perceptron algorithm, and then take the mean weight vector over all the history of learning instead of the last vector to make predictions: if the learning algorithm has done T iterations, the mean weight vector is $\frac{1}{T} \sum_{t=1}^{T} w_t$. Using an averaged perceptron avoids to overfit the training set.

3.4.2 Passive-aggressive algorithms

Here we review a class of learning algorithms similar to the perceptron in that they are online and update their weights in a similar way: the passive-aggressive (PA) algorithms (Crammer *et al.*, 2006). The key difference is passive-aggressive does not have a constant learning rate but rather adapts the size of the update step to satisfy a constraint: *large margin* classification. Maximizing the margin of a separation augments the ability of the model to generalize. More formally, given a new example (x_t, y_t) from the training set, the signed margin is defined as $y_t(w_t \cdot x_t)$. The purpose of PA learning is to achieve a classification of the example with margin at least 1. The hinge loss $l(w; (x_t, y_t)) := \max\{0, 1 - y_t(w_t \cdot x_t)\}$ is perfectly suited to this configuration: it is positive only if the margin is lower than 1 and non negative else where. Following the perceptron philosophy, we seek the minimum displacement of the current weight vector w_t to reach margin 1, which can be found by solving the following equation:

$$w_{t+1} = \underset{w \in \mathbb{R}}{\text{argmin}} \frac{1}{2} \|w - w_t\|^2 \quad \text{s.t.} \quad \max\{0, 1 - y_t(w_t \cdot x_t)\} = 0 \qquad \text{(PA)}$$

This optimization consists in obtaining a safe (i.e., with large margin) classification for the example that is currently learned. Fortunately, this is a convex optimisation problem that can be solved analytically using Lagrange multipliers[8]: $w_{t+1} = w_t + \tau_t y_t x_t$ with $\tau_t = \frac{l_t}{\|x_t\|^2}$. This has the real advantage to achieve an update in one step, avoiding numerical optimization to solve the problem.

A problem with PA is that enforcing a margin of at least 1 is somehow too brutal and may cause large shifts of the weight vector, hindering the global performances of the model. The solution introduced by Crammer *et al.* (2006) consists in introducing a parameter to tune the balance between the minimization of the displacement of the weight vector and the satisfaction of a large margin. Two versions, that can also be solved analytically with Lagrange multipliers under the Karush-Khun-Tucker conditions, are given[9]:

$$w_{t+1} = \underset{w \in \mathbb{R}}{\text{argmin}} \frac{1}{2} \|w - w_t\|^2 + C\xi \quad \text{s.t.} \quad l(w; (x_t, y_t)) \leq \xi \text{ and } \xi \geq 0 \qquad \text{(PA-I)}$$

$$w_{t+1} = \underset{w \in \mathbb{R}}{\text{argmin}} \frac{1}{2} \|w - w_t\|^2 + C\xi^2 \quad \text{s.t.} \quad l(w; (x_t, y_t)) \leq \xi \qquad \text{(PA-II)}$$

The step sizes for PA-I and PA-II are respectively $\tau_t = \min\left\{C, \frac{l_t}{\|x_t\|^2}\right\}$ and $\tau_t = \frac{l_t}{\|x_t\|^2 + \frac{1}{2C}}$. The complete learning procedure for PA is given in Algorithm 3.2.

Similarly to the bound on errors of the perceptron, a bound on cumulative squared loss can be exposed:

Theorem Crammer *et al.* (2006) $\mathcal{D} = (x_t, y_t)_{0 \leq t < N}$. Suppose that there exists $R > 0$ such that $\|x_t\| \leq R$ for all t, and $u \in \mathbb{R}^n$ such that $l(u; (x_t, y_t)) = 0$ for all t. Then $\sum_{t=0}^{N-1} l_t^2 \leq \|u\|^2 R^2$.

Now, as we wrote before, the simplest way to address multiclass learning is to learn one model per class. It can also be done in a more "passive-aggressive" way by, for each training instance, ranking the models with regards to their score on the instance, and update the models only if the one corresponding to the true class is not the first. Matsushima *et al.* (2010) proposed a method for determining more precisely which models should be updated. They introduce *support class*, which roughly corresponds to the minimum number of models to update to obtain a correct classification of the current instances. Again, there are tree versions of the update which are all calculated in closed form using Lagrange mutlipliers.

[8] See Boyd & Vandenberghe (2004, chapter 5) for an introduction to Langrange multipliers techniques.
[9] These versions are inspired from Vapnik (1998).

INPUT: Training data: $\mathcal{T} = \{(\boldsymbol{x}_i, P_i\}_{i=1}^{T}$, Iterations: N, Aggressiveness parameter: C
OUTPUT: Weight vectors $\boldsymbol{w}$

Initialize: $w_1 = (0, \ldots, 0)$
for $n : 1..N$ **do**
  for $t : 1..T$ **do**
    Receive instance: x_t
    Predict: $\hat{y}_t = \text{sign}(w_t \cdot x_t)$
    Receive correct label: $y_t \in \{-1, +1\}$
    Suffer loss: $l_t = \max\{0, 1 - y_t(w_t \cdot x_t)\}$
    Set step size:
      $\tau_t = \frac{l_t}{\|x_t\|^2}$ (PA)
      $\tau_t = \min\left\{C, \frac{l_t}{\|x_t\|^2}\right\}$ (PA-I)
      $\tau_t = \frac{l_t}{\|x_t\|^2 + \frac{1}{2C}}$ (PA-II)
    Update: $w_{t+1} = w_t + \tau_t y_t x_t$
  end for
end for

Figure 3.2: Passive-aggressive algorithms (PA, PA-I, PA-II)

3.4.3 Confidence-Weighted and Adaptive Regularization of Weights

NLP modeling typically involves a large number of features (there can be thousands or even millions, especially when using lexicalized features), most of them are observed rarely. However, the linear models presented above do not take into account this particularity of dealing with data sparseness in high dimension (the parameter can move too much when the model learns rare features). Dredze *et al.* (2008) introduced linear models that cope with such rare features by employing a measure of confidence in each weight: *Confidence-Weighed* (CW) models. A CW model has a less aggressive behavior when updating weights with strong confidence and it is more brutal with weights having low confidence. Formally, the confidence is modeled by a Gaussian distribution over the weight vector: $w \sim \mathcal{N}(\mu, \Sigma)$ where $\mu \in \mathbb{R}$ and Σ is a positive semi-definite matrix $d \times d$. To give an intuition, if the variance of a weight is high, the model is not confident in its value. On the contrary if it is narrow, the model is quite sure of the weight's value. Moreover, covariance (that is non diagonal values in Σ) captures information about feature interactions. Prediction is not achieved by picking up a weight vector at random from the distribution, which would involve a non deterministic behavior with some variance, but instead uses the mean μ. So the usage is the same as other linear models during prediction. The sole difference is in learning.

The learning procedure follows the same philosophy as PA algorithms, but instead of finding a minimum modification of weights such that the new learning instance is classified with a minimum margin, CW algorithm finds the closest Gaussian distribution over weights (i.e. finds new mean μ and covariance matrix Σ) to the current distribution such that the in-

stance is classified correctly with a minimal probability. Kullback-Leibler (KL) divergence is used to quantify the difference between the two distributions. Given a new labeled example (x_t, y_t), the equation to solve is:

$$(\mu_{t+1}, \Sigma_{t+1}) = \min_{\mu,\Sigma} D_{KL}\left(\mathcal{N}(\mu, \Sigma) \parallel \mathcal{N}(\mu_t, \Sigma_t)\right) \quad \text{s.t.} \quad \Pr(y_t(w \cdot x_t) \geq 0) \geq \eta$$

η is a parameter to control the aggressiveness of the learning procedure. Fortunately, the KL divergence between two Gaussian distributions can be calculated in closed form and the equation reduces to:

$$(\mu_{t+1}, \Sigma_{t+1}) = \min_{\mu,\Sigma} \frac{1}{2}\left\{\log\left(\frac{\det \Sigma_t}{\det \Sigma}\right) + \text{Tr}\left(\Sigma_t^{-1}\Sigma\right) + (\mu_t - \mu)^\top \Sigma_t^{-1}(\mu_t - \mu)\right\}$$
$$\text{s.t.} \quad y_t(\mu \cdot x_i) \geq \phi\sqrt{x_t^\top \Sigma x_t}$$

with number $\phi = \Phi^{-1}(\eta)$, where Φ is the cumulative distribution function of the standard normal distribution, i.e. $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-u^2} du$. Because of the square root, the constraint of that problem is not convex and the Lagrange multipliers technique cannot be directly employed to solve the problem analytically , so in the first version of CW, Dredze *et al.* (2008) linearized the constraint (i.e. removed the square root) to be able to solve the optimization. But Crammer *et al.* (2008) found that the problem could indeed be solved exactly with Lagrange multipliers and calculus by using a change of variables (they resort to square root decompostition of positive semi-definite matrices). However a last issue remains: when dealing with high dimensional data, the covariance matrix Σ can be very large and slow down the learning process. An easy solution to handle this is to approximate the matrix by its diagonal, and obtain an algorithmical complexity linear in the number of instances and the dimension. The full exact CW learning algorithm and its diagonal approximation will not be employed in this thesis (it is a direction to explore in future work), but can be found in Crammer *et al.* (2008).

The parameter a corresponds to the initial variance of the model, $\overrightarrow{0}$ is a d-dimensional vector filled with zeros and I_d is the unit diagonal matrix of size d. With $x_t = (x_t^1, \ldots, x_t^d)$, $\text{diag}^2(x_t)$ is the diagonal matrix with value x_t^i at (i, i). The formula are derived using Lagrange multipliers, second degree polynomial solutions and some linear algebra. The important fact to point out is the computation of α_t as a maximum. One should notice the similarity with PA algorithm: α_t is non zero only if the constraint $y_t(\mu \cdot x_i) \geq \phi\sqrt{x_t^\top \Sigma x_t}$ (or equivalently $\Pr(y_t(w \cdot x_t) \geq 0) \geq \eta$) is not satisfied. In other words, the update is achieved only if the instance classified with an unsatisfactory probability, i.e. less than a parameter η.

In practice, CW has shown rapid convergence of the learning procedure and state-of-the-art results on a number of text processing tasks compared to other online models, but also batch models such as SVM and maximum entropy model (Crammer *et al.*, 2012). A multiclass

version of CW was set up by Crammer *et al.* (2009a).

Even if reaching state-of-the-art performances, CW has shown to be very aggressive during learning which could result in overfitting the training data Crammer *et al.* (2008). Crammer *et al.* (2009b) proposed a modification of the model incorporating a regularization factor, the *Adaptive Regularization of Weights* (AROW). When iterating over the training examples, AROW adjusts its regularization which makes it more robust to noisy training data. Concretely, AROW still uses a Gaussian distribution over weights but replaces the objective function of CW by:

$$(\mu_{t+1}, \Sigma_{t+1}) = \min_{\mu,\Sigma} \left\{ D_{KL}\left(\mathcal{N}(\mu, \Sigma) \parallel \mathcal{N}(\mu_t, \Sigma_t)\right) + \lambda_1 l_{h^2}(y_t, \mu \cdot x_t) + \lambda_2 x_t^\top \Sigma x_t \right\}$$

where $l_{h^2}(y_t, \mu \cdot x_t) = \left(\max\left\{0, 1 - y_t(\mu \cdot x_t)\right\}\right)^2$ is the squared hinge loss, and λ_1 and λ_2 are parameters to control regularization and confidence. This objective is easier to solve analytically. To simplify the setting of the model, Crammer *et al.* (2009b) assume that $\lambda_1 = \lambda_2 = \frac{1}{2r}$ for $r > 0$.

We should notice that the condition for updating forces the model to change confidence Σ only if μ is also updated, i.e. when $\alpha_t > 0$. Again, a diagonal approximation of Σ reduces the algorithmic complexity of the model while still providing good performances. In practice, AROW performed better than CW on synthetic data and some real tasks (Crammer *et al.*, 2009b). Finally, there is a theoretical bound on the number of mistakes made by AROW:

Theorem (Crammer *et al.*, 2009b) Denote by $\mathcal{M}$ (resp. $\mathcal{U}$) the set of example indices for which AROW makes a mistake, i.e. $y_t(\mu_t \cdot x_t) \leq 0$ (resp. an update but no mistake, i.e. $0 < y_t(\mu_t \cdot x_t) \leq 1$). Let $X_\mathcal{M} = \sum_{t \in \mathcal{M}} x_t x_t^\top$, $X_\mathcal{U} = \sum_{t \in \mathcal{U}} x_t x_t^\top$ and $X_\mathcal{A} = X_\mathcal{M} + X_\mathcal{U}$. For any $u \in \mathbb{R}^d$,

$$|\mathcal{M}| \leq \sqrt{r\|u\|^2 + u^\top X_\mathcal{A} u} \sqrt{\log\left(\det\left(I_d + \frac{1}{r} X_\mathcal{A}\right)\right) + |\mathcal{U}|} + \sum_{t \in \mathcal{M} \cup \mathcal{U}} g_t - |\mathcal{U}|$$

where $g_t = \max\left\{0, 1 - y_t u^\top \cdot x_t\right\}$.

3.5 Learning with kernels

So far, we only discussed linear separation, emphasizing the fact that there exist fast online learning algorithms for linear models, some of which reaching state-of-the-art performances on some datasets. We now show how those algorithms can be used to learn non-linear predictors. A very popular technique is to resort to **kernels**. To briefly illustrate how it works, let us

underline that starting from a initial weight vector filled with zeros, all the learning algorithms above produce at the end a weight vector with is a linear combination of the training instances: $w = \sum_{t=0}^{N-1} \alpha_t x_t$. So a prediction with such weight vector is: $w \cdot x = \sum_{t=0}^{N-1} \alpha_t x_t \cdot x$. The essential idea for using kernels is to replace the canonical inner product $x \cdot x'$ by a positive definite kernel $\kappa(x, x')$ (Bakir *et al.*, 2007). A kernel function corresponds to an inner product in a space of much higher dimension (possibly infinite), such that $\kappa(x, x') = \phi(x) \cdot \phi(x')$, where ϕ is a non linear mapping from $\mathbb{R}^d$ to this space. A famous example of kernels is the *polynomial kernel* which has the form:

$$K(u, v) = (\alpha u \cdot v)^\beta$$

where α and β are parameters. To illustrate how much the corresponding mapping can be complex, we give ϕ in the case $\beta = 2$:

$$\phi(x) = \left(x_1^2, \ldots, x_n^2, \sqrt{2}x_1x_2, \ldots \sqrt{2}x_1x_n, \sqrt{2}x_2x_3, \ldots, \sqrt{2}x_{n-1}x_n, \sqrt{2c}x_1, \ldots, \sqrt{2c}x_n, c\right)$$

Predicting with a kernel, that is to say, using a linear predictor in the high dimension space is achieved as follows:

$$\text{sign}\left(w \cdot \phi(x)\right) = \text{sign}\left(\sum_{t=0}^{N-1} \alpha_t \kappa(x_t, x)\right)$$

So only the kernel function is needed (trying to compute ϕ directly may be very difficult or computationally inefficient). One way to reduce the complexity of the model is to restrain to the case where only a few α_t are non zeros. The corresponding vector are the so called *support vectors*. This class of models are called SVM classifiers (Cortes & Vapnik, 1995). Training a SVM amounts to solving a quadratic problem, this is too costly to have a reasonable runtime when applied to a large dataset with a high number of features. Another way to learn with kernels is to "kernelize" (i.e. replace inner product by kernel functions) the algorithms we presented above. But if the training set is a bit noisy, the number of support vectors might grow without bound and that can increase seriously the runtime and the amount of allocated memory. To cope with such problems, in the case of PA, (Wang & Vucetic, 2010) proposed modified "budgeted" algorithms that work with a fixed number of support vectors. These algorithms have a much more acceptable runtime and they were shown to achieve competitive results compared with the "non-budgeted" versions.

3.6 Learning to predict structures

In this section, we will see how linear models can be used for structured prediction (we are now interested by the learning procedures). In particular we will see that all the learning algorithms we detailed before (perceptron, PA, CW and AROW) apply to learning structures. Collins (2002) first introduced a method for learning to predict structures with a perceptron:

as explained in Subsection 3.3.3, an object $x \in \mathcal{X}$ and an output structure $y \in \mathcal{Y}$ are jointly represented by a feature vector $\phi(x, y)$. Let us recall that learning to predict structures can be seen as a classification problem with a very large number of categories, typically exponential in the size of the input. Using this representation, we seek the best structure prediction $\hat{y}(x) = \underset{y}{\text{argmax}}\,(w \cdot \Phi(x, y))$.

Computing the argmax is one of the challenging issues of structured learning: as mentioned in 3.3.3, the search space for structures is typically exponential in the size of the object x, and to be computationally efficient, the structure predictor should enumerate all the possible structures, but it should be coupled to a search algorithm. For example, Collins (2002) combined the *structured perceptron* with Viterbi algorithm to infer the best hidden sequence for POS tagging (in this case, x is a sequence of words and y a sequence of POS tags). For compatibility of the learning procedure with the search algorithm, the feature vector is decomposed in elementary feature vectors:

$$\Phi(x, y) = \sum_{i \in \mathcal{I}} \phi(x, y_i)$$

and thus the score of a structure is the sum of the scores of its sub-elements. In the case of Collins (2002), elementary units model the dependence between a current tag, some previous tags and the current word. The dependence $\phi(x, y_i)$ encompasses lexical features and n-grams. Another example is McDonald *et al.* (2005), where x is the words of a sentence and y, the structure to be computed, is a tree. The feature vector for the tree is decomposed in features over the edges of the tree, so that finding the best tree boils down to computing the scores for all possible edges and computing a maximum spanning tree on a complete weighted graph to find the argmax.

Now, let us focus on the learning procedure. The modified perceptron designed by Collins (2002) also cycles on the training instances (x, y) and updates its weight vector each time the predicted structure $\hat{y}$ is incorrect. The update is achieved only on symmetric difference of the sub-elements of $\Phi(x, y)$ and $\Phi(x, \hat{y})$: basically, after the update the algorithm will favour elements $\Phi(x, y_i)$ that were not selected in the prediction $\hat{y}$ and will avoid picking up sub-elements $\Phi(x, \hat{y_i})$ that were not part of the true structure y. This operation boils down to adding $\Phi(x, y) - \Phi(x, \hat{y})$ to the weight vector (possibly with a learning rate). The detailed learning procedure is given in algorithm 3.4. Again, a version of the theorem bounding the number of errors of the perceptron is given by Collins (2002). We do not report it here since the bound is exactly the same.

The learning method of algorithm 3.4 is called *prediction-based*, because it is the current prediction that is used when updating the weights. Another approach consists in employing a structure-specific loss l_s (which basically measures the difference between two structures) and trying to find the argmax that also maximizes this loss. This approach is called *max-loss* update[10]. The effect of max-loss is to find more non relevant sub-elements so that more wrong predictions will be pushed aside after the update.

[10] We the the terminology of Crammer *et al.* (2006).

INPUT: Training data: $\mathcal{T} = \{(\boldsymbol{x}_i, P_i\}_{i=1}^{T}$, Iterations: N, Learning rate: τ
OUTPUT: Weight vectors $\boldsymbol{w}$

Initialize: $w_1 = (0, \ldots, 0)$
for $n : 1..N$ **do**
  for $t : 1..T$ **do**
    Receive instance: x_t
    Predict: $\hat{y}_t = \underset{y \in \mathcal{Y}}{\text{argmax}}(w_t \cdot \Phi(x_t, y))$
    Receive correct structure: $y_t \in \mathcal{Y}$
    if $\hat{y}_t \neq y_t$ **then**
      Update: $w_{t+1} = w_t + \tau \left(\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t)\right)$
    else
      Proceed: $w_{t+1} = w_t$
    end if
  end for
end for

Figure 3.3: Structured perceptron learning (prediction-based).

Structured versions of PA Crammer *et al.* (2006), CW and AROW Mejer & Crammer (2010, 2011) have also been studied, showing results similar to or better than the perceptron. We do not write down the modifications of the algorithm since the only difference is to replace the vector x_t in binary classification by $\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t)$ for structured learning. We will go in further details in structured prediction when developing such models in Chapters 6, 7 and 8.

Kernelized versions of structured models can be defined in the SVM framework. For instance, see Yu & Joachims (2009) for initial work, and Ping *et al.* (2014) for an interesting extension to hidden variables.

3.7 Chapter summary

This chapter is an introduction to the learning techniques that we use in this thesis. Section 3.1 gives a small history of Machine Learning, briefly justifies using learning models instead of hand-made rules in many domains, and discusses the supervized and unsupervized learning paradigms. Section 3.2 draws up the problem of modelling objects with features to address a given task, and explains how all features (numbers and categorical) can be cast into a vectorial space. It then describes a few methods for selecting features. The first two (n-best selection, and greedy forward selection/backward elimination) are the most common and easy to implement. Section 3.3 explains how linear models can be used for binary classification, multiclass problems or structured predictions. It details the calculations with inner product between the weight vector (parameter of the model) and the feature vector (object to classify). Section 3.4 introduces online learning algorithms for training linear models. Online algorithms have the advantage of being memory and time efficient even though they may require several passes over the training set. Four kinds of learning algorithms are detailed (perceptron, PA, CW and

INPUT: Training data: $\mathcal{T} = \{(\boldsymbol{x}_i, P_i\}_{i=1}^{T}$, Iterations: N, Learning rate: τ, Loss: l_s
OUTPUT: Weight vectors $\boldsymbol{w}$

Initialize: $w_1 = (0, \ldots, 0)$
for $n : 1..N$ **do**
  for $t : 1..T$ **do**
    Receive instance: x_t
    Receive correct structure: $y_t \in \mathcal{Y}$
    Predict (max-loss): $\hat{y}_t = \underset{y \in \mathcal{Y}}{\text{argmax}}(w_t \cdot \Phi(x_t, y)) + \sqrt{l_s(y_t, y)}$
    if $\hat{y}_t \neq y_t$ **then**
      Update: $w_{t+1} = w_t + \tau\left(\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t)\right)$
    else
      Proceed: $w_{t+1} = w_t$
    end if
  end for
end for

Figure 3.4: Structured perceptron learning (max-loss).

AROW), and bounds on the number of errors they make during training are given as theorems. The full pseudo-code, including parameters is written for each algorithm we will use. Section 3.5 shows how the online learning algorithms can be combined with kernels to perform highly non-linear separations of data. Some references about optimizing the runtime when using kernels are given. Finally, Section 3.6 shows how the learning algorithms can be employed for structured prediction, insisting on the fact that in this case, the learning algorithm is often coupled to a search algorithm to compute the structure with the best score. This involves certain constraints on the feature representation of structures. Two modes for learning to predict structures are introduced: prediction-based and max-loss. Although kernels are not used in our experiments and although online large-margin models (PW, CW and AROW) do not represent the majority of our models, both provide a general framework in which all the models presented in this thesis would be extended for further optimizations.

Chapter 4

History and state of the art of coreference resolution

In this chapter, we introduce the various techniques that have been employed for coreference resolution. We begin by a brief history of anaphora/coreference resolution, in which we will see that the work on the subject really boomed in the 2000s, with more advanced resolution systems. Because the systems did not evolve in a unique direction, we will next attempt to do a presentation of the current resolution systems by topics, retaining the main characteristic of a system to put it into a category (sometimes, a system will appear in two categories).

Apart from this standalone state-of-the-art review, whose goal is to indicate the current trends in coreference resolution, more specific states of the art will be included in Chapters 5, 6 and 7 in their introductions and related work sections. In this way, it will be easier to make the connections between our proposals and previous work on the same topic.

4.1 Historical introduction

This section draws up a brief history of the study of coreference resolution, from the early discourse theoretical-based approaches to the recent empirical methods using less strict rules and parameterized through statistical analysis of corpora carefully annotated by specialists.

4.1.1 Early rule-based systems (1960-1990)

The early systems resolving anaphora[1], mostly restricted to processing pronouns or nouns having the same syntactic head as a previous noun (e.g. *"the price / this price"*), were introduced in the 1960s. STUDENT question-answering system (Bobrow, 1964) applied hardcoded deterministic rules to solve easy anaphora, but it was very limited. SHRDLU system (Winograd, 1972) was more elaborated as it employed more accurate heuristics based on syntax and a notion of focus favoring some potential antecedents over others. LUNAR system (Woods *et al.*, 1972) used a parser and a semantic interpreter (whereas the previous two systems only exploited basic lexical matching rules), however its heuristics were still very limited to some kinds of anaphora.

[1] See Mitkov (2002) or Poesio *et al.* (2011) for more details about the early developments of coreference resolution systems.

A few years later, the first *knowledge-based systems* appeared: at the time, the purpose was to implement different preference rules and constraints obtained from formal models and linguistic theories, which related anaphora behaviour to syntax, discourse and common knowledge. The earliest system based only on syntax was Hobbs' algorithm (Hobbs, 1976, 1978). It was a rule-based method relying on morphological information and the parse tree of sentences to resolve pronouns. Albeit based on syntactic constraints emanating from binding theory, the method was rather simplistic in the sense that it offered sufficient performances but could not improve on specific cases. However it is worth noticing that, three decades later, it was a variation of this algorithm that was implemented in the last step of the Stanford multi-sieves system (Lee *et al.*, 2011), the winner of CoNLL-2011[2].

The BFP algorithm for pronoun resolution (Brennan *et al.*, 1987; Walker, 1989), based on a dynamic framework extending centering theory (Grosz & Sidner, 1986), used a set of rules on discourse units (constraints on syntactic structure, types of referring expressions) providing a measure of *salience* to predict which entity is the most likely to be referred to at a given time (i.e position in the text). The algorithm proceeded in four steps: construct all acceptable anaphoric resolutions for a pronoun, filter them applying constraints, rank them using a transition model and select the best resolution. Despite its more elaborated foundations, the BFP algorithm showed results worse than those obtained with other rule-based systems of the same period. In particular, the much simpler Hobbs algorithm performed better (for instance, see benchmarks by Tetreault (1999)). Later, Strube (1998) proposed an alternative rule system to compute salience, providing improvements over the original algorithm.

One of the main problems during this period was that no common data existed to evaluate the resolution systems, and the methodology for scoring was neither standardized, and the numbers published did not reflect any absolute performance. Consequently, it was quite difficult to find out which method was more accurate.

4.1.2 Emergence of annotated corpora (1990s)

In the 1990s, because of the interest in applying coreference resolution to practical problems such as information extraction, a new methodology was introduced, more oriented toward empirical approaches to the matter. In particular, MUC-6 (muc, 1995) and MUC-7 (Chinchor, 1998) campaigns introduced coreference annotation schemes and provided corpora, which made it possible to compare all different resolution systems on a common basis, using the same metrics of evaluation[3]. Annotation schemes specified exactly what kinds of anaphora were taken into account. More precisely, MUC scheme was based on the annotation of *markables*, which encompass nouns, noun phrases and pronouns (demonstrative, personal and including possessives), and various relations between them including: bound anaphora, appositions, predicate nominals (in copulas), function and values (e.g. *"the temperature is 90°"*) and

[2]More details about this procedure are given in the following sections.
[3]See Section 2.3 of Chapter 2 for details on evaluation metrics.

metonymy[4].

Standard metrics to evaluate coreference resolution were subject of debate. The initial MUC metric was completed by others capturing different properties of the output (see Chapter 2 for more details about scoring metrics).

4.1.3 Data-driven approaches and machine learning (2000-present)

The availability of a considerable amount of annotated data and the late advances in machine learning made it possible to develop empirical methods for coreference resolution based on statistical learning techniques. The first modelling of the problem as mention pairs classification was introduced by (Soon *et al.*, 2001) and further developed by (Ng & Cardie, 2002b). This kind of systems incorporate a statistical model for pairs classification, and a heuristic procedure for making clusters. They are still competitive with current approaches.

Other systems were based on ranking instead of classification (Ng (2005)): in that framework, for each mention, the entities previously built are ranked and the mention is attached to the first in the ranking (or marked new). This approach is consistent with formal discourse theories in which the construction of references is achieved following the order of the text with backward links.

The next generation of data-driven systems intended, instead of relying on heuristic clustering or resolving anaphora in the order of the text, to make decision at the global level. Taking into account more complex interactions had also the drawback of introducing more combinatoric in the resolution systems. For instance, Denis & Baldridge (2007) had an integer linear programming formulation of constrains between the mentions in the text. This allows to explore a larger space of possibilities than when using heuristic clustering techniques, but the problem to solve is NP-complete in general. Other global approaches include first order logic modeling of mentions: Culotta *et al.* (2007) used graphical models, Poon & Domingos (2008) Markov logic networks (Richardson & Domingos, 2006).

At the beginning of the 2010s, CoNLL-2011 and 2012 Shared Tasks ((Pradhan *et al.*, 2011, 2012)) provided large corpora (in English, then Chinese and Arabic) with texts from various domains (news, conversion, biblical texts, web, etc), containing annotations of many kinds of coreference (including event coreference, see 2.4.2 for more details). But, even though the data was standardized, discussion about scoring metrics on detected mentions were engaged, finally leading to a reference implementation of the metrics Pradhan *et al.* (2014).

With 2011-12 Shared Tasks, a new kind of rule based systems appeared. Introduced by (Lee *et al.*, 2011), the multi-sieves outperformed classical machine learning approaches on

[4]Some of these relationships were dropped later by CoNLL annotation scheme: appositions and predicate nominals were not taken into account in evaluating coreference resolution anymore.

CoNLL 2011, using a set of cautious rules and patterns to establish coreference links, applying them from the highest to the lowest precision. Interestingly, the last step of this resolution method was to employ a version of Hobbs algorithm to resolve pronouns. Another sieve-based system incorporating parameters fitted on data (Chen & Ng, 2012) was quite competitive in CoNLL 2012 campaign. At the same time, (Fernandes *et al.*, 2012) used sieve-like rules for preprocessing mentions before applying machine learning techniques and were ranked first in the campaign with such a hybrid system.

4.2 State of the art in coreference resolution

In this section, we introduce a variety of empirical models and architectures for coreference resolution. It is difficult to define a proper classification of systems since their features can overlap, and it is also hard to obtain a ranking of current approaches because their performance depends on the choice of features for the underlying statistical models and the way text is preprocessed.

4.2.1 Pairwise models and decoding strategies

The “pairwise model" for coreference resolution is a very generic system, and quite an efficient one, since different evolutions of this basis scheme have reached state-of-the-art results. It should be viewed as a family of models having a common architecture. Basically, a pairwise model is a model for classifying pairs of mentions as coreferent or not, coupled with a rule-based procedure to create clusters of mentions called a “decoder". The role of the decoder is to create a coherent output, i.e. enforce transitivity between coreference links provided by the pairwise model. Such model does not ensure this property (as it considers every pair independently from each others). Since enough annotated data is provided now, pair classification models are usually based on statistical models[5]. A discussion of the features employed to represent the data can be found in 4.2.9.

Decoding strategies The most obvious way to obtain real clusters from the output of a pairwise model is the transitive closure on the graph of positively classified pairs and was employed by McCarthy & Lehnert (1995). In a seminal paper, Soon *et al.* (2001) introduced a range of features for *learning* a pairwise model, and a new way of decoding the output: the so called “closest-first" decoding consists in taking each mention *in the order of the text* and merging it (i.e. put in the same cluster) with the closest previous mention with which a positive coreference link is in the output. Contrarily to the transitive closure, this decoder removes links from the output of the pair model and selects only one backward link for each mention if it exits. This approach comes with a special way of sampling the data for the statistical model.

[5]See Chapter 3 for a presentation of some to these models.

Ng & Cardie (2002b) proposed a variant of this decoder, based on the confidence of the pair model. They chose to take the previous mention with which the confidence of the model was the highest compared to all other pairs formed with previous mentions and merge it with the examined mention. This is called “best-first" decoding[6]. They also introduced a different way of sampling training data to learn the pair model such that the pair model is more adapted to the decoder used downstream (see Uryupina (2004) for a another interesting sampling method for pairwise models).

Concerning the latest pairwise models, Stoyanov *et al.* (2010a) propose both closest-first and best-first strategies in their implementation, and Björkelund & Farkas (2012) have a mixed strategy, which is a closest first form for pronouns and a best-first for the mentions with another grammatical type.

Current implementations More recently, pairwise models have shown to be robust when employed with the most recent sets of features. Bengtson & Roth (2008) show that this simple architecture gives state of the art results, and that such systems can always be regarded as a strong baseline. Stoyanov *et al.* (2010a,b) propose a state of the art implementation of pairwise models, with different possibilities of statistical learners and decoders, and was shown to have median score on CoNLL-2011 shared task (Stoyanov *et al.* (2011)). Versley *et al.* (2008) propose a toolkit for coreference resolution based on pairwise models, which has been extended to process several languages (Broscheit *et al.* (2010)).

4.2.2 Local learning, global decoding

In order to make global coherent decisions from a pair model, Klenner (2007) reduces the problem of decoding to a constrained optimization problem: instead of having a greedy strategy to make clusters, they define an integer linear programming (ILP) problem such that transitivity must be ensured in the output, but also the sum of weights on the link must be maximized. Denis & Baldridge (2009) define a similar architecture, but take advantage of the ILP problem to include additional constraints such as anaphoricity and entity types. ILP makes it possible to define more complex and global decoding strategies, but it has a severe drawback: the optimization problem falls into the class of NP-complete problems, and unless one finds a way to drastically reduce the number of links to process (either by filtering or by dividing the problem into sub-problems), this approach does not scale well to larger documents. Contrarily, most state of the art resolution systems have an algorithmic complexity in $\mathcal{O}(N^2)$ where N is the number of mentions in the document. Indeed, most of them analyse the set of mention pairs, or mention subsets of lower size.

[6] In Chapter 6 we study the topological properties of the greedy decoders and quantify the effect of selecting links given a priority measure on them (e.g. best first backward link) with the transitive closure as referential.

4.2.3 Graph-cut methods

Another approach to decoding consists in separating clusters provided by the transitive closure into smaller and more reliable clusters. Nicolae & Nicolae (2006) first introduced the technique by using an intermediate “best graph cut" problem. Their method employ an additional model that, given a partition, decides to stop or continue cutting clusters. However, by trying to learn an optimal stopping criterion, they have to face the problem of sampling the (exponential number of) states of a partition to be able to create a learning data set. They achieved results comparable to what obtained by Ng & Cardie (2002b) (in MUC score).

Cai & Strube (2010a) propose a different approach based on hyper-graph partitioning. This model employs spectral clustering and learns how to partition the mentions (this approach can then also be classified in the previous subsection). Martschat *et al.* (2012) introduce a different graph method, based on a more elaborated decoder. It uses a set of various pairwise relations of two kinds: positive relations indicate that it is more likely to have a coreference link between the pair of mentions, and negative relations indicate otherwise. These are weighted by a statistical analysis of the training data (which amounts to learning a model), and passed to a greedy decoder, which basically balances positive and negative indicators to create coreference clusters. Later, Martschat (2013) shows that the statistical weighting of relations may not be useful at all, and that the system could reduce to a simple rule-based system.

4.2.4 Entity modelling and ranking

As opposed to pairwise models which only “see" characteristics of mentions (or pairs), other systems attempt to model cluster-mention relations, and cluster-cluster relations. The goal is to use the fact that, when merging mentions to make clusters by the decoding strategies above, at some point, a mention can be merged to a mention already merged with another mention. So, by taking into account all the information available about a partial entity, the system should be more accurate than if it examines only mention-mention relations. Luo *et al.* (2004) models the probability of a mention to a previously created entity. Moreover, it does not apply this model with a greedy decoding strategy, but employs a beam-search (which roughly amounts to looking at the decisions to be made several steps forward) to find better ways of selecting attachments of mentions to the current partial clusters.

Ng (2005) defines a completely different approach to the matter, by using a ranking model to find the previous partial cluster to which a mention is the most likely to be attached to (the mentions are taken in the order of the text). In a sense, modelling coreference as a ranking problem is closer to theoretical work on anaphora, where the attachment of an anaphor to a previous entity is seen as a sequential process, Denis & Baldridge (2008) propose specialized ranking models, adapting to the fact that the attachment of a mention to a previous entity depends on its grammatical type (see also Morton (2000) for similar considerations about

mention types). Rahman & Ng (2011) extend the ranking model by adding lexical features and integrating information about anaphoricity of mentions in the ranking process.

Haghighi & Klein (2010) model entity relations through sharing lexical information and entity types, which help their model to leverage semantic constraints. Durrett *et al.* (2013) define a model with factor graphs enforcing structural constrains on entities (some semantic properties of coreferent mentions must be the same). Stoyanov & Eisner (2012) propose a system that makes the easiest decisions first, and the next decisions by using cluster-level features. This latter approach is comparable to the multi-pass sieves (Lee *et al.* (2011)), except it does learn weights from training data.

4.2.5 Graphical models for global learning

Graphical models are well-suited for modelling a joint probability (from a generative perspective), but also to make global classification decision (from a discriminative point of view). Hence they can be employed to model the coreference problem globally, at the document level. McCallum & Wellner (2004) introduced the first discriminative approach with graphical models to coreference resolution, which made it possible to not consider pairs of mentions independently. Daumé III & Marcu (2005) introduced a clustering method with a Dirichlet process prior constraining the distribution of the size of the clusters (in particular, it ensures to get many small clusters and very few big ones). Haghighi & Klein (2007) propose a quite elaborated graphical model modelling entities, salience (for mention detection), and a dedicated modelling of pronouns. Culotta *et al.* (2007) Haghighi & Klein (2010) and Durrett *et al.* (2013) introduce models based on factor graphs to share information between entities and make more sensible decisions regarding the semantic types of entities. Finally, Poon & Domingos (2008) propose a coreference resolver based on Markov logic network (MLN). MLN can be regarded as frameworks to build graphical models, and make it possible to integrate logical constraints in the model. Similarly to ILP methods, MLN can easily integrate logical constraints such as semantic matches, but contrarily to ILP, it is not always specified for them to be exact on constraints, and inference can be faster. Song *et al.* (2012) use also MLN for their coreference resolution system. The set of features they have is quite different, though, and their learning procedure is completely supervised.

4.2.6 Latent tree models

As an alternative to global decoding via ILP, which casts the procedure in a too complex problem, and to graphical modelling, which generally requires a sampling procedure for inference, Yu & Joachims (2009) introduced a model representing the global architecture of clusters by latent trees. Basically, a cluster is viewed as a connected component of a graph and can be represented by a tree, the lightest structure connecting the elements of a connected component. From a discriminative perspective, the model learns to *build* such trees, given a document.

Fernandes *et al.* (2012), the winner of CoNLL-2012 Shared Task, proposed a different shape of latent trees derived from best-first decoding. They additionally filter out some of the mention pairs to get a more accurate learning of the weights. Chang *et al.* (2013) work on the same kind of latent trees and add rule-based constraints to improve the accuracy of the model. Finally, Björkelund & Kuhn (2014) define a more elaborated learning procedure that improves the latent tree model. Aside from the initial latent tree model which employed a structured SVM, the other systems are based on linear models, learned with a simple averaged perceptron algorithm in the case of Fernandes *et al.* (2012), showing that the benefits of structured modelling can be considerable. In Chapter 6, we provide a more thorough discussion about latent trees and the benefits of structured learning for coreference resolution.

4.2.7 Rule-based models

The first rule-based model was Hobbs algorithm (Hobbs, 1978). It was defined as a simple procedure to resolve pronouns, by making an extensive usage of parsing trees. Initially evaluated manually on 100 pronouns from three text genres (assuming that the text was perfectly parsed), with an accuracy of 88.3%, Hobbs' algorithm was later re-evaluated by several authors, obtaining results ranging from 76% to 82% (Matthews & Chodorow, 1988; Lappin & Leass, 1994; Tetreault, 2001). It is kind of a state of the art procedure since a very similar procedure is employed by the now famous multi-pass sieves system of Lee *et al.* (2011). That system was ranked first on CoNLL-2011 Shared Task, and was not machine learning based, contrarily to other participating systems. Basically sieves are a set of rules that builds entities iteratively and propagates constraints that prevent some mentions to be linked together. The major idea is to apply the rules from the most accurate to the less accurate (generally, less accurate rules do not apply everywhere because of the propagated constraints). A more detailed presentation of the sieves (Lee *et al.* (2013)) shows that a significative portion of the final score is due to pronoun resolution: an improvement of 10% in CoNLL score is reported by applying the last sieve, which is roughly a Hobbs algorithm. (Chen & Ng, 2012) proposed an extended version of the multi-pass sieves, by adding lexicalized features and statistically learned weights on the sieves. Another noticeable rule-based system is the multi-graph resolver due to Martschat (2013), which is an untrained version of Martschat *et al.* (2012) (all the weights are set to one). If these rather simple systems do not outperform machine learning based approaches now, they nonetheless point out the importance of an accurate preprocessing of the data (gender and number detection, named entity typing, parsing).

4.2.8 Unsupervised models

Aside from rule-based resolution systems, which to some extent can be considered as unsupervised models because they do not learn to discriminate from training data[7], there are a few

[7]The rules can be optimized to address coreference resolution for a particular class of document, though.

systems that learn to resolve coreference from un-annotated data. Such approaches can be useful when the coreference problem is addressed on a specific class of document for which no annotated data exists (because it is in a language for which no corpus exits, or because it is a domain not covered by current corpora). Haghighi & Klein (2007) first introduced an unsupervised system: their approach is to employ a non parametric Baysian model with many features for modelling cross document entities, salience to detect mentions, and a specific treatment of pronouns. The model has hyperparameters that have to be tuned on annotated data. Ng (2008) proposed an improvement of the model, using EM algorithm to set the parameters. These first unsupervised models were less accurate than discriminative (i.e. supervised) models, but the system of Poon & Domingos (2008), based on Markov logic network (which is also unsupervised) achieved state of the art results. It has not been compared to other systems on the latest corpora (CoNLL-2011 and 2012) and with the new evaluation metrics (Ceaf, BLANC).

4.2.9 Features for learning models

Except for the rule based models, all the systems presented above are machine learning based, and consequently require a good representation of data for the task (mention pairs, mention-entities, or simply mention classification). The choice and quality of features plays a crucial role in the architecture, improving the set of features often results in significant gains in the partition quality. This aspect of statistical resolver is also a problem when trying to replicate a system: usually features depend on the quality of preprocessing (like parsing, semantic type finding, morphological analysis, etc) for which there are no standard tools or software to use. Consequently differences between apparently identical system may be due to the way features are computed. In this section, we give pointers to relevant papers for defining or selecting features.

Classical hand-crafted features Soon *et al.* (2001) introduced an initial set of features that is now broadly used. These features are based on string matches, pronoun features, characterization of NP (definite, demonstrative), number and gender, semantic classes (like named entity types: person, organization, location, time, etc) and alias (e.g. “U.S." and “United States"). Ng & Cardie (2002b) improved some of theses features, and showed that significant improvement could be achieved through feature selection. Bengtson & Roth (2008) proposed a set of accurate features which made it possible to define a competitive pairwise model, showing again the importance of the choice of features. Noticeably, they use the output of an anaphoricity detector as a feature in the coreference model (Ng (2004b) showed previously that the propragation of errors of such anaphoricity detector used as a strong constraint on a pair model required a fine tuning of the system). Rahman & Ng (2011) give an exhaustive list of all the features they used for representing mention pair interactions and anaphoricity. A more detailed analysis of features for coreference resolution can be found in Recasens & Hovy (2009).

Lexicalized features Chen & Ng (2012) integrate lexical pairs in their system, which results in a high dimensional and sparse system, but help capturing more coreference interaction. Song *et al.* (2012) explicitly define the interaction between features and their action on coreference link through the first-order logic of MLN. Durrett & Klein (2013) propose a different approach: instead of employing hand-craft features, they make a different use of the data by mostly considering lexicalized features (boolean features indicating what is the head word, last word, previous word, etc, for a given mention). With tens of thousands of features, they manage to capture as much as hand-crafted features, with the noticeable exception of semantic features. They address this latter problem by completing their set of features with classical semantic features.

Feature selection Feature selection has been a very well studied topic in machine learning (see for instance Dash & Liu (1997); Kohavi & John (1997); Weston *et al.* (2000); Peng *et al.* (2005)). However, specific methods have been developed for coreference resolution: Uryupina & Poesio (2012) shows that domain specific modelling does improve the quality of a pairwise model. Ekbal *et al.* (2011); Uryupina *et al.* (2011b) define a method for optimizing the set of features with regards to the evaluation metrics, so that the whole system is tuned for improving the quality of partitions instead of that of the classification model. Fernandes *et al.* (2012) have a different approach: they use a procedure similar to entropy methods for creating classification trees to create products of features. However, it is only a procedure for expanding the space of features and not pruning it.

Lexical knowledge and world knowledge Representation of data is not just a technical problem concerning statistical models that can be solved by selection methods or feature products (or more generally by kernels). There is important work on representing linguistic knowledge, and more broadly world knowledge.

The lexical database Wordnet (Miller (1995)) has been used quite early in coreference resolution systems, but its lack of coverage on certain domains can reduce the advantage it provides elsewhere. Many authors tried to tackle the sparsity of such database by either extracting knowledge from the Web (see for instance Uryupina *et al.* (2011a)), or by converting the collaborative encyclopedia into a relational database (Ponzetto & Strube (2006)).

Bergsma & Lin (2006) defined a bootstrapping procedure to extract names gender and number from Google N-grams. These are the only data extracted from the Web that were authorized in CoNLL-2012 Shared Task Pradhan *et al.* (2012). Bergsma & Yarowsky (2011) proposed a tool for detecting non-referential pronouns *it* using the local context of the pronoun. The tool is based on Google N-grams, and runs very fast thanks to a good compression of data. It was shown to outperform state of the art tools in the domain.

4.3 Chapter summary

The first part of this chapter contains a brief history of automated coreference resolution, relating the empirical turn of the research due to the rise of information extraction and text processing in NLP. Since the 2000s, most coreference resolvers are statistical, mostly due to the fact that, in English, a considerable amount of annotated data is available.

The second part of the chapter sums up the state of the art for the empirical methods in coreference resolution. The problem has been modeled in various ways, as a mention pair classification problem coupled to a clustering procedure, as a ranking problem, and now more global approaches have emerged. The main advantage of global methods is that they enforce semantic coherence within clusters. A possible drawback is that they complexify the task from the computational point of view.

Chapter 5

Feature space hierarchy learning for pairwise coreference resolution

The results of this chapter have been published, first, as a long paper in TALN 2013 (Lassalle & Denis, 2013a), and an extended version of it as a long paper in ACL 2013 (Lassalle & Denis, 2013b).

As mentioned in our introductory chapter (Chapter 1), our first objective is to develop a framework for improving the representation of data in the statistical models we use. As we shall see thereafter, linear models are particularly popular and widely used in coreference resolution. To address our objective, we place ourselves in this linear framework, and try to get an improvement on current models by modifying the way features are implemented.

This chapter proposes a new method for significantly improving the performance of pairwise coreference models. Given a set of indicators, our method learns how to best separate types of mention pairs into equivalence classes for which we construct distinct classification models. In effect, our approach finds an optimal feature space (derived from a base feature set and indicator set) for discriminating coreferential mention pairs. Although our approach explores a very large space of possible feature spaces, it remains tractable by exploiting the structure of the hierarchies built from the indicators. Our experiments on the CoNLL-2012 Shared Task English datasets (gold mentions) indicate that our method is robust relative to different clustering strategies and evaluation metrics, showing large and consistent improvements over a single pairwise model using the same base features. Our best system obtains a competitive 67.2 *of average F1 over MUC,* B^3*, and CEAF which, despite its simplicity, places it above the mean score of other systems on these datasets.*

5.1 Introduction

Coreference resolution is the problem of partitioning a sequence of noun phrases (or *mentions*), as they occur in a natural language text, into a set of referential *entities*. A common approach to this problem is to separate it into two modules: on the one hand, one defines a model for evaluating coreference links, in general a discriminative classifier that detects coreferential mention pairs. On the other hand, one designs a method for grouping the detected links into

a coherent global output (i.e. a partition over the set of entity mentions). This second step is typically achieved using greedy heuristics McCarthy & Lehnert (1995); Soon *et al.* (2001); Ng & Cardie (2002b); Bengtson & Roth (2008), although more sophisticated clustering approaches have been used, too, such as graph cut methods Nicolae & Nicolae (2006); Cai & Strube (2010a) and Integer Linear Programming (ILP) formulations Klenner (2007); Denis & Baldridge (2009). Despite its simplicity, this two-step strategy remains competitive even when compared to more complex models utilizing a global loss Bengtson & Roth (2008).

In this kind of architecture, the performance of the entire coreference system strongly depends on the quality of the local pairwise classifier.[1] Consequently, a lot of research effort on coreference resolution has focused on trying to boost the performance of the pairwise classifier. Numerous studies are concerned with feature extraction, typically trying to enrich the classifier with more linguistic knowledge and/or more world knowledge Ng & Cardie (2002b); Kehler *et al.* (2004); Ponzetto & Strube (2006); Bengtson & Roth (2008); Versley *et al.* (2008); Uryupina *et al.* (2011a). A second line of work explores the use of distinct local models for different types of mentions, specifically for different types of *anaphoric* mentions based on their grammatical categories (such as pronouns, proper names, definite descriptions) Morton (2000); Ng (2005); Denis & Baldridge (2008).[2] An important justification for such specialized models is (psycho-)linguistics and comes from theoretical findings based on salience or accessibility Ariel (1988). It is worth noting that, from a machine learning point of view, this is related to feature extraction in that both approaches actually recast the pairwise classification problem in higher dimensional feature spaces.

In this chapter, we claim that mention pairs should not be processed by a single classifier, and instead should be handled through specific models. But we are furthermore interested in *learning* how to construct and select such differential models. Our argument is therefore based on statistical considerations, rather than on purely linguistic ones[3]. The main question we raise is, given a set of indicators (such as grammatical types, distance between two mentions, or named entity types), how to best partition the pool of mention pair examples in order to best discriminate coreferential pairs from non coreferential ones. In effect, we want to learn the "best" subspaces for our different models: that is, subspaces that are neither too coarse (i.e., unlikely to separate the data well) nor too specific (i.e., sensitive to data sparseness and noise). We will see that this is also equivalent to selecting a single large adequate feature space by using the data.

Our approach generalizes earlier approaches in important ways. For one thing, the definition of the different models is no longer restricted to grammatical typing (our model allows for various other types of indicators) or to the sole typing of the anaphoric mention (our models can also be specific to a particular type antecedent or to the two types of the mention pair).

[1] There are however no theoretical guarantees that improving pair classification will always result in overall improvements if the two modules are optimized independently.

[2] Sometimes, distinct sample selections are also adopted during the training of the distinct local models Ng & Cardie (2002b); Uryupina (2004).

[3] However it should be underlined that the statistical viewpoint is complementary to the linguistic work.

More importantly, we propose an original method for learning the best set of models that can be built from a given set of indicators and a training set. These models are organized in a hierarchy, wherein each leaf corresponds to a mutually disjoint subset of mention pair examples and the classifier that can be trained from it. Our models are trained using the *Online Passive-Aggressive* algorithm or *PA* Crammer *et al.* (2006), a large margin version of the Perceptron (see Chapter 3). Our method is exact in that it explores the full space of hierarchies (of size at least 2^{2^n}) definable on an indicator sequence, while remaining scalable by exploiting the particular structure of these hierarchies with dynamic programming. This approach also performs well, and it largely outperforms the single model. As will be shown based on a variety of experiments on the *CoNLL-2012 Shared Task* English datasets, these improvements are consistent across different evaluation metrics and for the most part independent of the clustering decoder that was used.

The rest of this chapter is organized as follows. Section 5.2 discusses the underlying statistical hypotheses of the standard pairwise model and defines a simple alternative framework that uses a simple separation of mention pairs based on grammatical types. Next, in Section 5.3, we generalize the method by introducing indicator hierarchies and explain how to learn the best models associated with them. Section 5.4 provides a brief system description and Section 5.5 evaluates the various models on CoNLL-2012 English datasets.

5.2 Modeling pairs

Pairwise models basically employ one local classifier to decide whether two mentions are coreferential or not. When using machine learning techniques, this involves certain assumptions about the statistical behavior of mention pairs.

5.2.1 Statistical assumptions

Let us adopt a probabilistic point of view to describe the prototype of pairwise models. Given a document, the number of mentions is fixed and each pair of mentions follows a certain distribution (that we partly observe in a feature space). The basic idea of pairwise models is to consider mention pairs independently from each other (that is why a decoder is necessary to enforce transitivity).

If we use a single classifier to process all pairs, then they are supposed to be identically distributed. We claim that pairs should not be processed by a single classifier because they are not identically distributed (or a least the distribution is too complex for the classifier); rather, we should separate different "types" of pairs and create a specific model for each of them.

Separating different kinds of pairs and handling them with different specific models can lead to more accurate global models. For instance, some coreference resolution systems

process different kinds of anaphors separately, which suggests for example that pairs containing an anaphoric pronoun behave differently from pairs with non-pronominal anaphors. One could rely on a rich set of features to capture complex distributions, but here we actually have a rather limited set of elementary features (see Section 5.4) and, for instance, using products of features must be done carefully to avoid introducing noise in the model. Instead of imposing heuristic product of features, we will show that a clever separation of instances leads to significant improvements of the pairwise model.

5.2.2 Feature spaces

5.2.2.1 Definitions

We first introduce the problem more formally. Every pair of mentions m_i and m_j is modeled by a random variable:

$$\begin{array}{rcl} P_{ij} : \Omega & \rightarrow & \mathcal{X} \times \mathcal{Y} \\ \omega & \mapsto & (x_{ij}(\omega), y_{ij}(\omega)) \end{array}$$

where Ω classically represents randomness, $\mathcal{X}$ is the space of objects ("mention pairs") that is not directly observable and $y_{ij}(\omega) \in \mathcal{Y} = \{+1, -1\}$ are the labels indicating whether m_i and m_j are coreferential or not. To lighten the notations, we will not always write the index ij. Now we define a mapping:

$$\begin{array}{rcl} \phi_{\mathcal{F}} : \mathcal{X} & \rightarrow & \mathcal{F} \\ x & \mapsto & \mathbf{x} \end{array}$$

that casts pairs into a feature space $\mathcal{F}$ through which we observe them. For us, $\mathcal{F}$ is simply a vector space over $\mathbb{R}$ (in our case many features are Boolean; they are cast into $\mathbb{R}$ as 0 and 1).

For technical coherence, we assume that $\phi_{\mathcal{F}_1}(x(\omega))$ and $\phi_{\mathcal{F}_2}(x(\omega))$ have the same values when projected on the feature space $\mathcal{F}_1 \cap \mathcal{F}_2$: it means that common features from two feature spaces have the same values.

From this formal point of view, the task of coreference resolution consists in fixing $\phi_{\mathcal{F}}$, observing labeled samples $\{(\phi_{\mathcal{F}}(x), y)_t\}_{t \in TrainSet}$ and, given partially observed new variables $\{(\phi_{\mathcal{F}}(x))_t\}_{t \in TestSet}$, recovering the corresponding values of y.

5.2.2.2 Formalizing the statistical assumptions

We claimed before that all mention pairs seemed not to be identically distributed since, for example, pronouns do not behave like nominals. We can formulate this more rigorously: since

the object space $\mathcal{X}$ is not directly observable, we do not know its complexity. In particular, when using a mapping to a too small feature space, the classifier cannot capture the distribution very well: the data is too noisy.

Now if we say that pronominal anaphora do not behave like other anaphora, we distinguish two kinds of pairs: we state that the distribution of pairs in $\mathcal{X}$ is a mixture of two distributions, and we deterministically separate pairs to their specific distribution part. In this way, we may separate positive and negative pairs more easily if we cast each kind of pair into a specific feature space. Let us call these feature spaces $\mathcal{F}_1$ and $\mathcal{F}_2$. We can either create two independent classifiers on $\mathcal{F}_1$ and $\mathcal{F}_2$ to process each kind of pair or define a single model on a larger feature space $\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2$. If the model is linear (which is our case), these approaches happen to be equivalent.

So we can actually assume that the random variables P_{ij} are identically distributed, but drawn from a complex mixture. A new issue arises: we need to find a mapping $\phi_{\mathcal{F}}$ that renders the best view on the distribution of the data.

From a theoretical viewpoint, the higher the dimension of the feature space (imagine taking the direct sum of all feature spaces), the more we get details on the distribution of mention pairs and the more we can expect to separate positives and negatives accurately. In practice, we have to cope with data sparsity: there will not be enough data to properly train a linear model on such a space. Finally, we seek a feature space situated between the two extremes of a space that is too big (sparseness) or too small (noisy data). The core of this work is to define a general method for choosing the most adequate space $\mathcal{F}$ among a huge number of possibilities when we do not know *a priori* which is the best.

5.2.2.3 Linear models

In this work, we try to linearly separate positive and negative instances in the large space $\mathcal{F}$ with the *Online Passive-Aggressive* (PA) algorithm (Crammer *et al.* (2006)): the model learns a parameter vector $\mathbf{w}$ that defines a hyperplane that cuts the space into two parts. The predicted class of a pair x with feature vector $\phi_{\mathcal{F}}(x)$ is given by:

$$C_{\mathcal{F}}(x) := sign(\mathbf{w}^T \cdot \phi_{\mathcal{F}}(x))$$

Linearity implies an equivalence between: (i) separating instances of two types, t_1 and t_2, in two independent models with respective feature spaces $\mathcal{F}_1$ and $\mathcal{F}_2$ and parameters $\mathbf{w}^1$ and $\mathbf{w}^2$, and (ii) a single model on $\mathcal{F}_1 \oplus \mathcal{F}_2$. To see why, let us define the map:

$$\phi_{\mathcal{F}_1 \oplus \mathcal{F}_2}(x) := \begin{cases} \begin{pmatrix} \phi_{\mathcal{F}_1}(x)^T & 0 \end{pmatrix}^T & \text{if } x \text{ is typed } t_1 \\ \begin{pmatrix} 0 & \phi_{\mathcal{F}_2}(x)^T \end{pmatrix}^T & \text{if } x \text{ is typed } t_2 \end{cases}$$

and the parameter vector $\mathbf{w} = \begin{pmatrix} \mathbf{w}^1 \\ \mathbf{w}^2 \end{pmatrix} \in \mathcal{F}_1 \oplus \mathcal{F}_2$. Then we have:

$$C_{\mathcal{F}_1 \oplus \mathcal{F}_2}(x) = \begin{cases} C_{\mathcal{F}_1}(x) & \text{if } x \text{ is typed } t_1 \\ C_{\mathcal{F}_2}(x) & \text{if } x \text{ is typed } t_2 \end{cases}$$

Now we check that the same property applies when the PA fits its parameter $\mathbf{w}$. For each new instance of the training set, the weight is updated according to the following rule[4]:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathcal{F}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 \ \ s.t.\ l(\mathbf{w}; (x_t, y_t)) = 0$$

where $l(\mathbf{w}; (x_t, y_t)) = \min(0, 1 - y_t(\mathbf{w} \cdot \phi_{\mathcal{F}}(x_t)))$, so that when $\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2$, the minimum if x is typed t_1 is $\mathbf{w}_{t+1} = \begin{pmatrix} \mathbf{w}^1_{t+1} \\ \mathbf{w}^2_t \end{pmatrix}$ and if x is typed t_2 is $\mathbf{w}_{t+1} = \begin{pmatrix} \mathbf{w}^1_t \\ \mathbf{w}^2_{t+1} \end{pmatrix}$ where the $\mathbf{w}^i_{t+1}$ corresponds to the updates in space $\mathcal{F}_i$ independently from the rest. This result can be extended easily to the case of n feature spaces. Thus, with a deterministic separation of the data, a large model can be learned using smaller independent models.

5.2.3 An example: separation by *gramtype*

To motivate our approach, we first introduce a simple separation of mention pairs which creates 9 models obtained by considering all possible pairs of grammatical types {*nominal, name, pronoun*} for *both* mentions in the pair (a similar fine-grained separation can be found in Chen *et al.* (2011)). This is equivalent to using 9 different feature spaces $\mathcal{F}_1, \ldots, \mathcal{F}_9$ to capture the global distribution of pairs. With the PA, this is also a single model with feature space $\mathcal{F} = \mathcal{F}_1 \oplus \cdots \oplus \mathcal{F}_9$. We will call it the GRAMTYPE model.

As we will see in Section 5.5, these separated models significantly outperform a single model that uses the same base feature set. But we would like to define a method that adapts a feature space to the data by choosing the most adequate separation of pairs.

5.3 Hierarchizing feature spaces

In this section, we have to keep in mind that separating the pairs in different models is the same as building a large feature space in which the parameter $\mathbf{w}$ can be learned by parts in independent subspaces.

[4]The parameter is updated to obtain a margin of a least 1. It does not change if the instance is already correctly classified with such margin.

5.3.1 Indicators on pairs

For establishing a structure on feature spaces, we use *indicators* which are deterministic functions on mention pairs with a small number of outputs. Indicators classify pairs in predefined categories in one-to-one correspondence with independent feature spaces. We can reuse some features of the system as indicators, e.g. the grammatical or named entity types. We can also employ functions that are not used as features, e.g. the approximate position of one of the mentions in the text.

The small number of outputs of an indicator is required for practical reasons: if a category of pairs is too refined, the associated feature space will suffer from data sparsity. Accordingly, distance-based indicators must be approximated by coarse histograms. In our experiments the outputs never exceeded a dozen values. One way to reduce the output span of an indicator is to binarize it like binarizing a tree (there many possible binarizations). This operation produces a hierarchy of indicators which is exactly the structure we exploit in what follows.

5.3.2 Hierarchies for separating pairs

We define hierarchies as combinations of indicators creating finer categories of mention pairs: given a finite sequence of indicators, a mention pair is classified by applying the indicators successively, each time refining a category into subcategories, just like in a decision tree (each node having the same number of children as the number of outputs of its indicator). We allow the classification to stop before applying the last indicator, but the behavior must be the same for all the instances. So a hierarchy is basically a sub-tree of the complete decision tree that contains copies of the same indicator at each level.

If all the leaves of the decision tree have the same depth, this corresponds to taking the Cartesian product of outputs of all indicators for indexing the categories. In that case, we refer to *product-hierarchies*. The GRAMTYPE model can be seen as a two level product-hierarchy (figure 5.1).

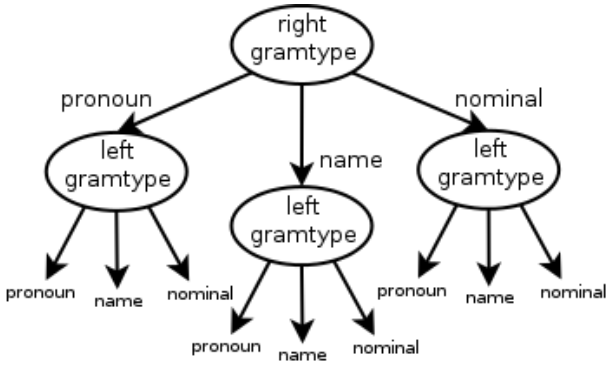

Figure 5.1: GRAMTYPE seen as a product-hierarchy

Product-hierarchies will be the starting point of our method to find a feature space that fits the data.

Now choosing a relevant sequence of indicators should be achieved through linguistic intuitions and theoretical work (*gramtype* separation is one of them). The system will find by itself the best usage of the indicators when optimizing the hierarchy. The sequence is a parameter of the model.

5.3.3 Relation with feature spaces

Like we did for the GRAMTYPE model, we associate a feature space $\mathcal{F}_i$ to each leaf of a hierarchy. Likewise, the sum $\mathcal{F} = \bigoplus_i \mathcal{F}_i$ defines a large feature space. The corresponding parameter $\mathbf{w}$ of the model can be obtained by learning the $\mathbf{w}_i$ in $\mathcal{F}_i$.

Given a sequence of indicators, the number of different hierarchies we can define is equal to the number of sub-trees of the complete decision tree (each non-leaf node having all its children). The minimal case is when all indicators are Boolean. The number of full binary trees of height at most n can be computed by the following recursion: $T(1) = 1$ and $T(n+1) = 1 + T(n)^2$. So $T(n) \geq 2^{2^n}$: even with small values of n, the number of different hierarchies (or large feature spaces) definable with a sequence of indicators is gigantic (e.g. $T(10) \approx 3.8.10^{90}$).

Among all the possibilities for a large feature space, many are irrelevant because for them the data is too sparse or too noisy in some subspaces. We need a general method for finding an adequate space without enumerating and testing each of them.

5.3.4 Optimizing hierarchies

Let us assume now that the sequence of indicators is fixed, and let n be its length. To find the best feature space among a very high number of possibilities, we need a criterion we can apply without too much additional computation. For that we only evaluate the feature space locally on pairs, i.e. without applying a decoder on the output. We employ 3 measures on pairwise classification results: precision, recall and F1-score. Now selecting the best space for one of these measures can be achieved by using dynamic programming techniques. In the rest of the paper, we will optimize the F1-score.

Training the hierarchy Starting from the product-hierarchy, we associate a classifier and its proper feature space to each node of the tree[5]. The classifiers are then trained as follows:

[5] In the experiments, the classifiers use a copy of a same feature space, but not the same data, which corresponds to crossing the features with the categories of the decision tree.

for each instance there is a unique path from the root to a leaf of the complete tree. Each classifier situated on the path is updated with this instance. The number of iterations of the Passive-Aggressive is fixed.

Computing scores After training, we test all the classifiers on another set of pairs[6]. Again, a classifier is tested on an instance only if it is situated on the path from the root to the leaf associated with the instance. We obtain TP/FP/FN numbers[7] on pair classifications that are sufficient to compute the F1-score. As for training, the data on which a classifier at a given node is evaluated is the same as the union of all data used to evaluate the classifiers corresponding to the children of this node. Thus we are able to compare the scores obtained at a node to the "union of the scores" obtained at its children.

Cutting down the hierarchy For the moment we have a complete tree with a classifier at each node. We use a dynamic programming technique to compute the best hierarchy by cutting this tree and only keeping classifiers situated at the leaf. The algorithm assembles the best local models (or feature spaces) together to create larger models. It goes from the leaves to the root and cuts the sub-tree starting at a node whenever it does not provide a better score than the node itself, or on the contrary propagates the score of the sub-tree when there is an improvement. The details are given in algorithm 5.2. This algorithm solves an argmax problem: finding the sub-hierarchy with which pair classification has the highest F1-score on a development set.

```
1:  list ← list of nodes given by breadth-first search
2:  for node in reverse(list) do
3:      if node.children ≠ ∅ then
4:          if sum-score(node.children) > node.score then
5:              node.TP/FP/FN ← sum.num(node.children)
6:          else
7:              node.children ← ∅
8:          end if
9:      end if
10: end for
```

Figure 5.2: Cutting down a hierarchy

Let us briefly discuss the correctness and complexity of the algorithm. Each node is seen two times so the time complexity is linear in the number of nodes which is at least $\mathcal{O}(2^n)$. However, only nodes that have encountered at least one training instance are useful and there are $\mathcal{O}(n \times k)$ such nodes (where k the size of the training set). So we can optimize the algorithm

[6]The training set is cut into two parts, for training and testing the hierarchy. We used 10-fold cross-validation in our experiments.

[7]True positives, false positives and false negatives.

to run in time $\mathcal{O}(n \times k)$[8]. If we scan the list obtained by breadth-first search backwards, we are ensured that every node will be processed after its children. (*node.children*) is the set of children of *node*, and (*node.score*) its score. *sum-num* provides TP/FP/FN by simply adding those of the children and *sum-score* computes the score based on these new TP/FP/FN numbers. (line 7) cuts the children of a node when they are not used in the best score. The algorithm thus propagates the best scores from the leaves to the root which finally gives a single score corresponding to the best hierarchy. Only the leaves used to compute the best score are kept, and they define the best hierarchy.

Relation between cutting and the global feature space We can see the operation of cutting as replacing a group of subspaces by a single subspace in the sum (see figure 5.3). So cutting down the product-hierarchy amounts to reducing the global initial feature space in an optimal way.

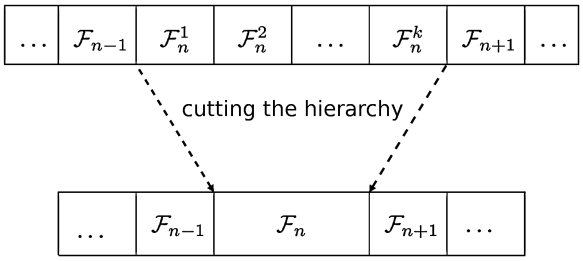

Figure 5.3: Cutting down the hierarchy reduces the feature space

To sum up, the whole procedure is equivalent to training more than $\mathcal{O}(2^n)$ Perceptrons simultaneously and selecting the best performing one.

5.4 System description

Our system consists in the pairwise model obtained by cutting a hierarchy (the PA with selected feature space) and using a greedy decoder to create clusters from the output. It is parametrized by the choice of the initial sequence of indicators.

5.4.1 The base features

We used classical features that can be found in details in Bengtson & Roth (2008) and Rahman & Ng (2011): grammatical type and subtype of mentions, string match and substring, apposition and copula, distance (number of separating mentions/sentences/words), gender/number

[8] In our experiments, cutting down the hierarchy was achieved very quickly, and the total training time was about five times longer than with a single model.

match, synonymy/hypernym and animacy (using WordNet), family name (based on lists), named entity types, syntactic features (gold parse) and anaphoricity detection.

5.4.2 Indicators

As indicators we used: left and right grammatical types and subtypes, entity types, a boolean indicating if the mentions are in the same sentence, and a very coarse histogram of distance in terms of sentences. We systematically included right gramtype and left gramtype in the sequences and added other indicators, producing sequences of different lengths. The parameter was optimized by document categories using a development set *after* decoding the output of the pairwise model.

5.4.3 Decoders

We tested 3 classical greedy link selection strategies that form clusters from the classifier decision: Closest-First (merge mentions with their closest coreferent mention on the left) Soon *et al.* (2001), Best-first (merge mentions with the mention on the left having the highest positive score) Ng & Cardie (2002b); Bengtson & Roth (2008), and Aggressive-Merge (transitive closure on positive pairs) McCarthy & Lehnert (1995). Each of these decoders is typically (although not always) used in tandem with a specific sampling selection at training. Thus, Closest-First for instance is used in combination with a sample selection that generates training instances only for the mentions that occur between the closest antecedent and the anaphor Soon *et al.* (2001).

	P	R	F1
SINGLE MODEL	22.28	**63.50**	32.99
RIGHT-TYPE	29.31	45.23	35.58
GRAMTYPE	39.12	45.83	42.21
BEST HIERARCHY	**45.27**	51.98	**48.40**

Table 5.1: Pairwise scores on CoNLL-2012 test.

5.5 Experiments

	MUC			B^3			CEAF			
Closest-First	P	R	F1	P	R	F1	P	R	F1	Mean
SOON	79.49	**93.72**	**86.02**	26.23	**89.43**	40.56	**49.74**	19.92	28.44	51.67
SINGLE MODEL	78.95	75.15	77.00	51.88	68.42	59.01	37.79	43.89	40.61	58.87
RIGHT-TYPE	79.36	67.57	72.99	69.43	56.78	62.47	41.17	**61.66**	49.37	61.61
GRAMTYPE	80.50	71.12	75.52	66.39	61.04	63.6	43.11	59.93	50.15	63.09
BEST HIERARCHY	**83.23**	73.72	78.19	**73.50**	67.09	**70.15**	47.30	60.89	**53.24**	**67.19**

	MUC			B^3			CEAF			
Best-First	P	R	F1	P	R	F1	P	R	F1	Mean
NGCARDIE	**81.02**	**93.82**	**86.95**	23.33	**93.92**	37.37	**40.31**	18.97	25.80	50.04
SINGLE MODEL	79.22	73.75	76.39	40.93	75.48	53.08	30.52	37.59	33.69	54.39
RIGHT-TYPE	77.13	65.09	70.60	48.11	66.21	55.73	31.07	47.30	37.50	54.61
GRAMTYPE	77.21	65.89	71.10	49.77	67.19	57.18	32.08	**47.83**	38.41	55.56
BEST HIERARCHY	78.11	69.82	73.73	**53.62**	70.86	**61.05**	35.04	46.67	**40.03**	**58.27**

	MUC			B^3			CEAF			
Aggressive-Merge	P	R	F1	P	R	F1	P	R	F1	Mean
SINGLE MODEL	83.15	88.65	85.81	35.67	**88.18**	50.79	36.3	28.27	31.78	56.13
RIGHT-TYPE	**83.48**	**89.79**	**86.52**	36.82	88.08	51.93	45.30	33.84	38.74	59.07
GRAMTYPE	83.12	84.27	83.69	44.73	81.58	57.78	45.02	42.94	43.95	61.81
BEST HIERARCHY	83.26	85.2	84.22	**45.65**	82.48	**58.77**	**46.28**	**43.13**	**44.65**	**62.55**

Table 5.2: CoNLL-2012 test (gold mentions): Closest-First, Best-First and Aggressive-Merge decoders.

5.5.1 Data

We evaluated the system on the English part of the corpus provided in the *CoNLL-2012 Shared Task* Pradhan *et al.* (2012), referred to as CoNLL-2012 here. The corpus contains 7 categories of documents (over 2K documents, 1.3M words). We used the official train/dev/test data sets. We evaluated our system in the *closed mode* which requires that only provided data is used.

5.5.2 Settings

Our baselines are a SINGLE MODEL, the GRAMTYPE model (section 5.2) and a RIGHT-TYPE model, defined as the first level of the *gramtype* product hierarchy (i.e. grammatical type of the anaphora Morton (2000)), with each greedy decoder and also the original sampling with a single model associated with those decoders.

The hierarchies were trained with 10-fold cross-validation on the training set (the hierarchies are cut after cumulating the scores obtained by cross-validation) and their parameters are optimized *by document category* on the development set: the sequence of indicators obtaining the best average score *after* decoding was selected as parameter for the category. The obtained hierarchy is referred to as the BEST HIERARCHY in the results. We fixed the number of iterations for the PA for all models.

In our experiments, we consider only the *gold mentions*. This is a rather idealized setting but our focus is on comparing various pairwise local models rather than on building a full coreference resolution system. Also, we wanted to avoid having to consider too many parameters in our experiments.

5.5.3 Evaluation metrics

We use the three metrics that were used in CoNLL-2012 Shared Task evaluation[9], namely: **MUC** (Vilain *et al.*, 1995), $\mathbf{B}^3$ (Bagga & Baldwin, 1998a) and **CEAF** with the ϕ_4 similarity function (Luo, 2005). In addition, we report the CoNLL score which we recall is an unweighted average over the F1 scores given by the three metrics. Following common practice, we use micro-averaging when reporting our scores for entire datasets. For more information on the metrics, we refer to Chapter 2.

[9]BLANC metric Recasens & Hovy (2011) results are not reported since they are not used to compute the CoNLL-2012 global score. However we can mention that in our experiments, using hierarchies had a positive effect similar to what was observed on B^3 and CEAF.

5.5.4 Results

The results obtained by the system are reported in Table 5.2. The original sampling for the single model associated to Closest-First and Best-First decoder are referred to as SOON and NGCARDIE.

The P/R/F1 pairwise scores before decoding are given in table 5.1. BEST HIERARCHY obtains a strong improvement in F1 (+15), a better precision and a less significant diminution of recall compared to GRAMTYPE and RIGHT-TYPE.

Despite the use of greedy decoders, we observe a large positive effect of pair separation in the pairwise models on the outputs. On the mean score, the use of distinct models versus a single model yields F1 increases from 6.4 up to 8.3 depending on the decoder. Irrespective of the decoder being used, GRAMTYPE always outperforms RIGHT-TYPE and single model and is always outperformed by BEST HIERARCHY model.

Interestingly, we see that the increment in pairwise and global score are not 100% correlated: for instance, the strong improvement of F1 between RIGHT-TYPE and GRAMTYPE results in a small increase of the global score.

Depending on the document category, we found some variations as to which hierarchy was learned in each setting, but we noticed that parameters starting with right and left gramtypes often produced quite good hierarchies: for instance right gramtype $\rightarrow$ left gramtype $\rightarrow$ same sentence $\rightarrow$ right named entity type.

We observed that product-hierarchies did not perform well without cutting (especially when using longer sequences of indicators, because of data sparsity) and could obtain scores lower than the single model. Hopefully, after cutting them the results always became better as the resulting hierarchy was more balanced.

Looking at the different metrics, we notice that overall, pair separation improves B^3 and CEAF (but not always MUC) after decoding the output: GRAMTYPE provides a better mean score than the single model, and BEST HIERARCHY gives the highest B^3, CEAF and mean score.

The best classifier-decoder combination reaches a score of 67.19, which would place it above the mean score (66.41) of the systems that took part in the *CoNLL-2012 Shared Task* (gold mentions track). Except for the first at 77.22, the best performing systems have a score around 68-69. Considering the simple decoding strategy we employed, our current system sets up a strong baseline.

5.6 Conclusion and outlooks

In this chapter, we described a method for selecting a feature space among a very large number of choices by using linearity and by combining indicators to separate the instances. We employed dynamic programming on hierarchies of indicators to compute the feature space providing the best pairwise classifications efficiently. We applied this method to optimize the pairwise model of a coreference resolution system. Using different kinds of greedy decoders, we showed a significant improvement of the system.

Our approach is flexible in that we can use a variety of indicators. In the future, we will apply the hierarchies on finer feature spaces to make more accurate optimizations. Observing that the general method of cutting down hierarchies is not restricted to modeling mention pairs, but can be applied to problems having Boolean aspects. A straightforward extension of this work should then be to study how such methods can be employed to address other tasks in computational linguistics (e.g. anaphoricity detection or discourse and temporal relation classification wherein position information may help separating the data). This work would be, however, beyond the scope of this thesis.

In this work, we have only considered standard, heuristic linking strategies like Closest-First. So, a natural extension of this work is to combine our method for learning pairwise models with more sophisticated decoding strategies (like *Bestcut* or using ILP). Then we can test the impact of hierarchies with more realistic settings.

Finally, the method for cutting hierarchies should be compared to more general but similar methods, for instance polynomial kernels for SVM and tree-based methods (Hastie *et al.*, 2005). We also plan to extend our method by breaking the symmetry of our hierarchies. Instead of cutting product-hierarchies, we will employ usual techniques to build decision trees[10] and apply our cutting method on their structure. The objective is twofold: first, we will get rid of the sequence of indicators as parameter. Second, we will avoid fragmentation or overfitting (which can arise with classification trees) by deriving an optimal large margin linear model from the tree structure.

In the next chapters, we will use only simple hierarchies discussed here. One reason is that we will develop structured models for which it is not trivial to derive a hierarchy cutting procedure. However, because we noticed that features spaces produced by cutting on pairwise linear models also improve structured models, it is worth employing simple versions of hiearchies in this case (see chapter 6 and 7).

[10] Bansal & Klein (2012) show good performances of decision trees on coreference resolution.

Chapter 6

Learning Constrained Latent Structures for Coreference Resolution: a Comparative Approach

In the previous chapter we define a technique to improve linear models. In the experiments we carried out, we implemented pairwise models that only address coreference resolution locally, and require a decoder to create clusters. The issue with such models is that we learn to build coreference clusters only from local information. Our second objective (see chapter 1) is to investigate how to represent documents (or at least the coreference structure) with a single structure, which could be learned directly instead of learning local coreference links. The advantage of structured models is that they allow the system to make a global decision on coreference clusters rather than many local decisions that can be difficult to reconcile after.

This chapter compares several methods for learning latent structures encoding coreference clusters that optionally take into account very accurate constraints on mention pairs. We study the relationship between standard decoding strategies used with pairwise models and those used with structured learning of latent structures, providing both topological and empirical comparisons. In general, we show that sparse link selection strategies provide better results than those given by denser latent structures. In particular, Best-First decoding appears to outperform all other kinds of link selections, such as Maximum Spanning Tree and other intermediate decoders. We also show that further gains can be obtained by the addition of pairwise constraints. In this case, our model learns to complete a partial structure on a non-complete graph with mentions as vertices. Our experiments on the CoNLL-2012 dataset show that our best system obtains state-of-the-art results, and significant gains compared to standard locally-trained models.

6.1 Introduction

Recall that coreference resolution consists in partitioning *mentions* (typically noun phrases) occurring in a text into referential *entities*. Various models exist for addressing this task, from purely rule-based methods (e.g., the Stanford multi-pass sieve Lee *et al.* (2011)) to advanced machine learning techniques. Technically, coreference resolution can be seen as a clustering task, but many machine learning approaches since Soon *et al.* (2001) have reduced the problem of partitioning mentions to performing clustering on a weighted mention graph. Initial methods

proceeded by first inducing a local classifier to detect coreferring pairs of mentions and then applying a greedy heuristic decoder to create entities. Apart from the obvious transitive closure rule McCarthy & Lehnert (1995), more conservative decoders such as CLOSESTFIRST (Soon *et al.*, 2001) or BESTFIRST (Ng & Cardie, 2002b; Bengtson & Roth, 2008) have been used.

Recently, more global approaches were introduced, basically still working on a graph of mention pairs (i.e., using the same pairwise features), but with more elaborated models. In particular, Yu & Joachims (2009) represent mention clusters by latent trees using structural SVM in learning (a cluster is represented by a tree connecting its members). Fernandes *et al.* (2012) propose a related approach by learning to build latent trees with a structured large-margin perceptron (Collins, 2002). Structured latent tree methods are also investigated by Chang *et al.* (2012, 2013). This later approach is augmented with domain knowledge-based constraints. Common to these three approaches is that they recast the coreference resolution problem as a structured classification task (thereby folding the clustering objective inside the learning phase), and that they rely on latent tree-like structures for representing clusters. These approaches however differ in the type of decoder they use for deriving these latent structures, and ultimately in the shapes of these structures. Thus, Yu & Joachims (2009) resort to a Maximum Spanning Tree (MST, or MSF in this chapter) algorithm to produce their latent structures, while Chang *et al.* (2012) and Chang *et al.* (2013) use a standard BESTFIRST decoding. Interestingly, Fernandes *et al.* (2012), the winner of CoNLL-2012 Shared Task, also use a MST algorithm but on a weighted *directed* graph, which in effect amounts to running a BESTFIRST decoder on the corresponding undirected graph. These differences raise a number of interesting questions, the first being: What is the most relevant type of latent structure for learning coreference resolution? And, in turn: How does one characterize these structures formally, and how do they compare to the structures produced by the standard decoders used with mention pair models?

This chapter proposes a unified theoretical and experimental framework for comparing structured latent tree coreference systems. By studying the topological properties of the different decoders, we first demonstrate that MSF decoding is equivalent to transitive closure on the graph of *positive mention pairs* in terms of the clusterings it produces. Second, we show that from a topological and combinatoric point of view, the structures produced by MSF are much more expressive than the structures generated by BESTFIRST or CLOSESTFIRST (which are equally expressive). By contrast, our experiments on CoNLL-2012 reveal that empirically BESTFIRST outperforms MSF-based decoding, both with locally-trained and structured models. In general, our results indicate that sparse link selection methods lead to better performance than methods that rely on denser structures. What is remarkable is that, compared to other possible sparse link selection strategies, BESTFIRST still performs best. This result is arrived at by considering a family of decoders obtained by adding supplementary positive links to the BESTFIRST structure, thus converging to transitive closure. We also develop constrained versions of our structured models, incorporating pairwise must-link and cannot-link constraints. These constrained models learn to complete a partial structure given by must-link rules by respecting cannot-link constraints to obtain a latent representation of clusters. To sum up, our

comparison is done along three main directions: local vs. structured models, sparse vs. dense latent structure, and unconstrained vs. constrained decoding. This comparison relies on the same learning framework, namely the structured perceptron (Collins, 2002; Fernandes *et al.*, 2012), which makes our approach easily replicable.

The rest is organized as follows. Section 6.2 investigates the structural and topological properties of decoders. Section 6.3 introduces our structured models. Section 6.4 then details the features we employed and how the data is preprocessed. Section 6.5 details our three main experiments.

6.2 Decoding the weighted pair graph

A variety of machine learning-based coreference resolvers rely solely on modeling mention pairs. They use a feature representation of mention pairs, a statistical model to weight the graph of pairs, and a heuristic procedure or *decoder* to create clusters. In such systems, the decoder works as a link selector: it builds a subgraph of the weighted graph whose connected components are the clusters in the output partition. Here, we study the three decoders which underlie many standard mention pair models: CLOSESTFIRST iterates the mentions and links each one to the closest preceding mention with which the pair is classified positively (if it exists) (Soon *et al.*, 2001). BESTFIRST also iterates the previous mentions but links the current mention to the preceding mention for which the mention pair has the highest positive score (Ng & Cardie, 2002b; Bengtson & Roth, 2008). Finally, AGGRESSIVEMERGE, is a mere transitive closure on positive pairs. More recently, the MSF decoder was used in the context of latent tree models Yu & Joachims (2009).

In this section, we study these decoders in terms of their topological (i.e., the types of cluster they output) and structural (i.e., the set of links they select) properties. The main results are the finer cluster produced by BESTFIRST and CLOSESTFIRST compared to transitive closure, the equivalence of MSF decoding and transitive closure, and the compared combinatoric of MSF and BESTFIRST.

6.2.1 From decoders to structures

Any coreference model based on pairs of mentions can be represented in the following general framework. For a given document, we define a complete weighted undirected graph $G = (V, E, \omega)$, where V represents the (ordered) mentions[1] in the text, E all the mention pairs, and $\omega : E \to \mathbb{R}$ a weighting function indicating our belief in a pair to be coreferent. At this point, it does not matter how the weights are produced, this will be the subject of Section 6.3. Let G^+ denotes G from which negatively weighted edges have been removed. It may no longer

[1] We call either *vertex* or *mention* the same object.

be a connected graph. More importantly, it is sufficient for the three decoders to work on G^+. Now focusing on the selected links, we call *structures* the subgraph of G^+ made of the links provided by decoders. The straightforward question we can ask is: what are the possible structures for a decoder?

BESTFIRST Structures We denote by $\mathcal{S}^{best}(G^+)$ the set of all possible BESTFIRST structures on G^+ regardless of the values of the weights. This is a collection of forests: indeed, no cycle can be created by BESTFIRST linking. The proof is obvious: if there is a cycle, consider the rightmost vertex of the cycle (in the order of the text). Necessarily, two edges of the cycle have an end at this vertex which contradicts the fact that BESTFIRST chooses at most one backward link for each mention. The set of CLOSESTFIRST structures is also $\mathcal{S}^{best}(G^+)$, as one simply has to change the weights to obtain the same structures on a graph. More specifically, these two decoders output a *spanning forest* for G, that is a set of disjoint spanning trees covering all mentions, each tree corresponding to exactly one cluster. In practice, the structure effectively computed by BESTFIRST is:

$$\hat{F} = \underset{F \in \mathcal{S}^{best}(G^+)}{\arg\max} \sum_{e \in E_F} \omega(e)$$

where E_F denotes the edges of the forest F. The process of building a forest-like structure on G^+ with BESTFIRST is illustrated in figure 6.1: first the graph edges are weighted (using the score of a statistical model), negatively weighted edges are removed and clustering rules are applied. Final clusters are usually smaller than the connected component of the graph of positively weighted edges.

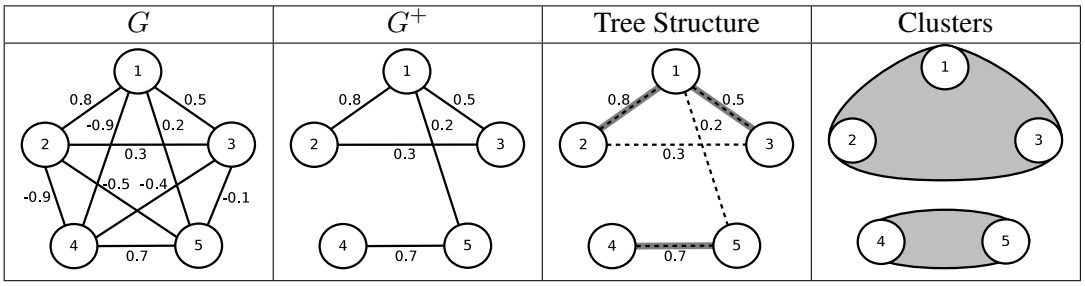

Figure 6.1: The different step of decoding a weighted graph using BESTFIRST to create clusters.

Maximum Spanning Forests We would like to characterize AGGRESSIVEMERGE structures within the same class of forest-like structures. To do this, we define an equivalent decoder, MSF (for Maximum Spanning Forest)[2], producing exactly the same partition given a weighted graph: it computes a maximum spanning forest on G^+ (recall that G^+ is not necessarily connected) using Kruskal algorithm, and links the mentions according to the selected edges. More

[2]This kind of structures was first used for coreference resolution by Yu & Joachims (2009).

precisely, if $\mathcal{S}^{span}(G^+)$ is the set of spanning forests for G^+, the computed structure is:

$$\tilde{F} = \underset{F \in \mathcal{S}^{span}(G^+)}{\arg\max} \sum_{e \in E_F} \omega(e)$$

Two well-known greedy algorithms for finding that optimal structure are Prim's algorithm (Prim, 1957) and Kruskal's algorithm (Kruskal, 1956); both have been shown to be correct and run in $\mathcal{O}(|E| \log(|V|))$ using an implementation with binary heap. Let us now give a proof of the equivalence of AGGRESSIVEMERGE and MSF. If we consider a connected component of G^+, by definition, a spanning tree on this component connects all vertices. A maximum spanning forest is a collection of spanning trees, with exactly one tree connected the elements of each connected component: indeed, if a tree is strictly contained in a component, that is there is another tree within the same component, it is possible to create a single bigger tree by adding a positive link (we work on G^+) between the two trees. By this operation we obtain a spanning forest with a greater weight, which contradicts the fact MSF has a maximum weight. Thus, if we link altogether the mentions in a same spanning tree of the forest, we exactly obtain the transitive closure of G^+. It is important to understand that coreference resolvers that use AGGRESSIVEMERGE or MSF yield exactly the same partitions. The only difference lies in the intermediate structure that MSF generates, and this will be crucial for learning.

6.2.2 Topological and structural properties

So far, we have shown that CLOSESTFIRST, BESTFIRST and MSF produce the sparsest possible structures to represent clusters. We now study more precisely the shape of these sparse structures and how they are related to the final partition of mentions. Which of these structures are more suitable for learning will be discussed in Section 6.5.

The topologies of partitions From the point of view of the clusters, CLOSESTFIRST and BESTFIRST produce partitions that are always refined version of that of MSF. This property is immediate by looking at the structures: $\mathcal{S}^{best}(G^+)$is a set of spanning forests, thus necessarily a subset of $\mathcal{S}^{span}(G^+)$, which contains all possible spanning forest on G^+. All trees in each structure are contained in a connected component of G^+, with the difference that one tree of the MSF matches exactly one component. Thus the clusters of CLOSESTFIRST and BESTFIRST are contained in the clusters of MSF (see Figure 6.2). Intuitively, a finer partition is more conservative as it is more precise on coreference links, but has a lower recall. This is illustrated in section 6.5, where we define intermediate decoders to see the differences between BESTFIRST and MSF.

The lack of expressiveness of MSF As we saw earlier, an essential topological property of MSF is that it only builds forests whose trees match exactly one connected component of G^+ and is strictly equivalent to a transitive closure in the partitions it produces. Therefore,

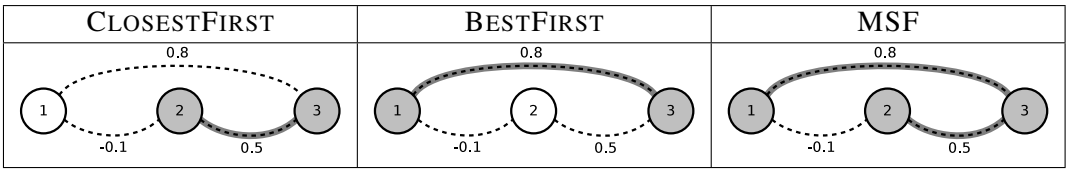

Figure 6.2: In general, CLOSESTFIRST and BESTFIRST produce a subpartition of that of MSF/ AGGRESSIVEMERGE.

regardless of the weights, the partition created by MSF only depends on the topology of G^+. The fact that it does not depend on the weights implies that MSF does not have much expressiveness in the sense that a small variation of the weights that does not alter the topology of G^+ does not result in a change in the output structure (while it could change the BESTFIRST's output)[3].

Combinatoric expressiveness of the structures Despite the lack of expressiveness discussed above, MSF structures have a combinatoric advantage over the other structures. Suppose we do not know the weighting function ω *a priori*, what structures can be expected to be built? We saw that $\mathcal{S}^{best}(G^+)$was included in $\mathcal{S}^{span}(G^+)$. However the converse is not true: as we can see on figure 6.2, the MSF tree cannot be generated by the two other decoders. Now the question is: how much bigger is $\mathcal{S}^{span}(G^+)$?

We can quantify the difference of combinatoric complexity of the structures (i.e. the number of configurations they can generate): without knowing the weights *a priori*, let us count the potential structures that can be built by each decoder. If n is the number of mentions (or vertices in the graph), CLOSESTFIRST and BESTFIRST link every mention independently to exactly one previous mention or no mention at all, which gives $n!$ possible configurations. As for MSF, Cayley's formula gives us the number of spanning trees for n vertices on a complete graph: n^{n-2}. If we examine the ratio of these numbers, using Stirling's formula we get:

$$\frac{n!}{n^{n-2}} \sim \frac{n^n e^n \sqrt{2\pi n}}{n^{n-2}} = n^{\frac{5}{2}} e^{-n} \sqrt{2\pi}$$

For a document with 10 mentions, it is 3.6%, and with 100 mentions, less than 10^{-38}. So compared to the others, the MSF decoder is far more expressive in the variety of the structures it can produce when having (potentially large) variation of the weights.

[3]The same property holds for CLOSESTFIRST.

6.2.3 Constraining the weighted graph

The framework developed so far makes it easy to integrate pairwise constraints. Given the linguistic properties of mentions, some potential coreferent links can be invalidated by *cannot-link* constraints (e.g., gender or number mismtaches). Conversely, some accurate patterns, or *must-link* constraints, can detect pairs that are very likely to be coreferent (e.g. matching strings for proper names). These constraints can be directly added to the graph: *cannot-links* are taken into account by removing the corresponding edges of G^+ so that no structure can select them, and *must-links* are integrated as initially selected edges for the structures[4]. Note that *cannot-links* as defined here are not hard constraints in the sense that a cluster may contain a *cannot-link* at the end. This approach gives more power to the learning model, which can actually cancel a *cannot-link*.

6.2.4 The structures of recent coreference resolvers

Finally, we can use our framework to analyse the different structures provided by different state-of-the-art coreference resolution systems. The discussion of how we learn ω is delayed until section 6.3.

Many recent systems use structures produced by the three decoders above: for instance, Bengtson & Roth (2008) use BESTFIRST, Stoyanov *et al.* (2010a) propose CLOSESTFIRST and BESTFIRST strategies, while Björkelund & Farkas (2012) add a mixed strategy depending on the grammatical type of the current mention. Chang *et al.* (2012, 2013) consider best backward links (i.e., BESTFIRST) in a latent structure with *must-link* and *cannot-link* constraints, Yu & Joachims (2009) use latent MSF, which we now know is equivalent to an AGGRESSIVEMERGE strategy, but in a structured learning model.

Fernandes *et al.* (2012) compute latent MST on a directed graph with Chu-Liu-Edmonds Algorithm (Chu & Liu, 1965; Edmonds, 1965), with filters for canceling out some edges beforehand (equivalent to *cannot-links*). Interestingly, since their graph is restricted to left-to-right edges, and because in a directed tree a node has at most one parent, in the maximum spanning tree they build, each mention is linked to the previous best (or none) in such way that the resulting structure is a BESTFIRST structure. Furthermore, it appears then that it is not necessary to use Chu-Liu-Edmonds Algorithm, because best link selection is sufficient to get the same structure.

Some of the models above use a *root*: an additional vertex linked with all other vertices with weight 0 (and kept in G^+). As a result, structures are always connected, but they encode the same clusters as if there were no root. A root is specifically employed by Fernandes *et al.* (2012) to tune a specific loss on root-mention links for the learning model.

[4]The resulting structures may contain cycles if *must-links* themselves create those cycles.

6.3 Learning latent structures

We now turn to the issue of learning the weights for scoring the different forest structures presented above. Following previous recent approaches (Yu & Joachims, 2009; Fernandes *et al.*, 2012; Chang *et al.*, 2013), we recast the problem of learning coreference clusterings as a structured classification problem.

6.3.1 From local to global learning

Local mention pair models are hampered by the fact that their weights are not related to the structures output by the decoder. In particular, they may not be suited to produce the most coherent structures. Structured learning, on the other hand, makes it possible to learn directly to build the structures and improve their quality by a global update on the document itself, and not only locally on pairs.

Directly learning partitions is difficult since we cannot compute the exact best partition in a tractable way. Enumerating partitions is unimaginable considering that the size of their space is exponential (equal to the Bell number). Sparse structures are well-suited to this approach because they are directly related to an optimization algorithm (both BESTFIRST and MSF compute an *argmax*).

6.3.2 Structured learning

The task of learning a clustering function is recast as a supervised structured classification problem. That is, the learning algorithm observes a set of examples $\mathcal{T} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{T}$, where $\boldsymbol{x}_i$ is an instance from a structured input space $\mathcal{X}$ (the space of documents) and $\boldsymbol{y}_i$ is a *structured* label from an output space $\mathcal{Y}$ whose size is exponential in the size of $\boldsymbol{x}$. Assume we have at hand an *embedding function* $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space equipped with inner product $\langle \cdot, \cdot \rangle$. Given a document $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ with n mentions, K_n denotes the weighted complete graph K_n on mentions. Let $\boldsymbol{y}$ be a structure on K_n whose set of edges is E_y. We restrict Φ to the following decomposed form:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(m,m') \in E_y} \phi(\boldsymbol{x}, m, m')$$

where $\phi(\boldsymbol{x}, m, m')$ is an embedding of the mention pair (m, m') depending on information contained in document $\boldsymbol{x}$ (e.g. a feature representation). If we additionally assume that we have at hand a fixed vector $\boldsymbol{w} \in \mathcal{H}$. The score associated to $(\boldsymbol{x}, \boldsymbol{y})$ is given by

$$\langle \boldsymbol{w}, \Phi(\boldsymbol{x}, \boldsymbol{y}) \rangle = \sum_{(m,m') \in E_y} \langle \boldsymbol{w}, \phi(\boldsymbol{x}, m, m') \rangle$$

In this case, the weighting function on G (the graph associated to document $\boldsymbol{x}$) employed in the previous section is then given by $\omega(m, m') = \langle \boldsymbol{w}, \phi(\boldsymbol{x}, m, m') \rangle$. As we shall see, $\boldsymbol{w}$ is a prediction vector that we will have to *learn* given documents labeled with their corresponding coreference clusters. To predict $\boldsymbol{x}$'s structure, we simply compute

$$F(\boldsymbol{x}) = \underset{F \in \mathcal{S}(K_n^+)}{\arg\max} \sum_{(m,m') \in E_F} \langle \boldsymbol{w}, \Phi(m, m') \rangle$$

where $\mathcal{S}(K_n^+)$ can be either $\mathcal{S}^{best}(K_n^+)$ or $\mathcal{S}^{span}(K_n^+)$ according to the kind of structures we choose (MSF or BESTFIRST). Since CLOSESTFIRST structures are not computed by maximizing their total weight (but only according to the topology of K_n^+), we do not define a structured learning procedure for them.

The crucial problem we are now faced with is that of learning a relevant weight vector $\boldsymbol{w}$. Indeed, we do not directly observe *gold* structures corresponding to a partition, and moreover, several structures can correspond to the same partition. Thus, we must create these target structures from the sole observation of clusters. Thereafter, we address this problem within a structured perceptron learning approach.

6.3.3 Latent structure perceptron-based learning

In the present case, $\mathcal{H} = \mathbb{R}^m$ with the canonical dot product. The goal of learning is to acquire the weight vector $\boldsymbol{w} \in \mathbb{R}^m$. Our parameter estimation relies on the online algorithm presented below.

INPUT: Training data: $\mathcal{T} = \{(\boldsymbol{x}_i, P_i\}_{i=1}^T$
OUTPUT: Weight vector $\boldsymbol{w}$

$\boldsymbol{w}^{(0)} = 0; \boldsymbol{v} = 0; i = 0$
for $n : 1..N$ **do**
  for $t : 1..T$ **do**
    Compute *true* label $y_i^{\boldsymbol{w}}$ from P_i and $\boldsymbol{w}^{(i)}$:
    $y_i^{\boldsymbol{w}} = \underset{y \in \tilde{\mathcal{Y}}_i}{\arg\max} \langle \boldsymbol{w}^{(i)}, \Phi(x_i, y) \rangle$
    Compute *max-loss* prediction $\tilde{y}$:
    $\tilde{y} = \underset{y \in \mathcal{Y}}{\arg\max} \langle \boldsymbol{w}^{(i)}, \Delta(x_i, y, y_i^{\boldsymbol{w}}) \rangle + l(y, y_i^{\boldsymbol{w}})$
    $\boldsymbol{w}^{(i+1)} = \boldsymbol{w}^{(i)} + \Delta(x_i, y_i^{\boldsymbol{w}}, \tilde{y})$
    $v = v + \boldsymbol{w}^{(i+1)}$
    $i = i + 1$
  end for
end for
$\boldsymbol{w} = \frac{\boldsymbol{v}}{(N \times T)}$

Figure 6.3: Structured perceptron learning algorithm with parameter averaging, in *max-loss* mode.

Note that $\Delta(x, y, y')$ stands for $\Phi(x, y) - \Phi(x, y')$, and $l(y, y')$ for the non-negative *loss* between labels y and y'. This algorithm is only given a sequence $\{\boldsymbol{x}_i, P_i\}_{i=1}^T$ of documents without their corresponding structures, but containing information on the partition of their

mentions. It starts with an initial weight vector $\boldsymbol{w}^{(0)}$ and iterates N times over the training examples, resulting in $N \times T$ iterations. At each round i, a *true* structure is computed in replacement of *gold* labels according to the current document $\boldsymbol{x}_i$ by finding the structure corresponding to the clustering with the best current weight (in the algorithm, $\tilde{\mathcal{Y}}_i$ corresponds to the structures compatible with the partition P_i). In practice, it is easily computed by removing non-coreferent edges from the complete graph according to the *gold* clustering, and then by choosing the structure on this graph with maximum weight (it is like working on G^+).

Next, we compute a predicted structure in *max-loss* mode (Crammer *et al.*, 2006).[5] This means that we use a trade-off between the maximization of the structure weight and finding a structure "far from the *true* structure", according to a loss counting the number of edges the structures do not have in common (the approach is similar to Fernandes *et al.* (2012)). The weight vector is updated by the difference of *true* and *max-loss* predicted labels in a structured perceptron manner. The final weight vector is obtained by averaging over the weight vectors obtained after each round. Weight averaging is common in online learning for it helps reducing overfitting (Freund & Schapire, 1999; Collins, 2002).

6.3.4 Constrained learning

As noted in section 6.2, we want to constrain our structures so that they obey some *must-link* and *cannot-link* rules. We proceed exactly as for inference, meaning that we constrain the structure construction by removing edges from the graph and adding must-link edges to the tree-shaped predicted structures. Integrating constraints in the learning procedure amounts to defining a model that *learns* how to complete a partial structure given by constraints, which is different from applying constraints only during coreference resolution.

6.4 System description

Our different coreference systems are based on pairwise representation of mention pairs, meaning we re-employ classical features. We define a *baseline* system, referred to as "local model" which is a simple pairwise model trained using an averaged Perceptron, associated to the three decoders we presented in section 6.2. As opposed to the local model, we set up a "global model", which consists in learning BESTFIRST and MSF structures with the algorithm given in section 6.3. For the global model we define constrained and unconstrained versions, and we add a supplementary "dense" global model which does not learning using tree-like structures but the whole weighted graph or its transitive closure (more details given in section 6.5).

[5] The more direct *prediction-based* (i.e., without loss) update always gave lower results in our development experiments, so we do not detail this learning mode here.

6.4.1 Feature set

Our system uses a classical set of features used for mention pair classification (for more details see Bengtson & Roth (2008); Rahman & Ng (2011)): grammatical type and subtypes, string and substring match, apposition and copula, distance (number of separating mentions/sentences/words), gender and number match, synonymy/hypernymy and animacy, family name (based on lists), named entity types, syntactic features (gold parse) and anaphoricity detection. We have an additional morphological feature which indicates if a verb is derived from a noun, to take into account coreference of events. In addition, we expand the feature space by using products of the above features with grammatical types. Recall that in Chapter 5, we defined a method for improving pairwise models by expanding the space of features. We found in our development that this technique also significantly improved the results of all models defined here. For the sake of clarity and replicability, we do not integrate the full hierarchy discovery in these experiments. We use the "pair of gramtypes" hierarchy instead, which yet provides sufficient improvement to the base models.

6.4.2 Constraints

We explored a reduced set of constraints, although sufficient to provide significant performance gains. Our first set of *must-links* is provided by our implementation of sieve 1 of Lee *et al.* (2011), which among other things accurately matches patterns involving the speaker of sentences (e.g. *He* said: "*I* believe you"). Another second set of *must-links* come from exact string matches of proper nouns. We also use several sets of *cannot-links*, coming from number, gender, and (un)animated mismatches, as well as i-within-i constraints.

6.5 Experiments

The objective of our experiments is to determine which model, local or global, sparse or dense, unconstrained or constrained performs the best on coreference resolution. All the models are implemented within the same framework (i.e. averaged perceptron on the same set of features) allowing for a sound comparison.

6.5.1 Experimental setup

Data We evaluated the models on the CoNLL-2012 Shared Task English CorpusPradhan *et al.* (2012), which has 7 categories of documents (more than 2K documents). We used the official Train/Dev/Test data sets. We evaluated our system in the *closed mode* which requires that only provided data is used (with the exception of data from WordNet and Bergsma & Lin (2006)'s gender and number data, which are are authorized). Finally, note that we

considered only the *gold mentions*. Our main objective is on comparing various competing learning-decoding combinations rather than on building a full end-to-end coreference resolution system.

Evaluation We compare our various systems using the three popular coreference resolution metrics: MUC (Vilain *et al.*, 1995), B^3 (Bagga & Baldwin, 1998a), and Entity-based CEAF (or $CEAF_e$) (Luo, 2005). Following (Pradhan *et al.*, 2012), we also report a global F1-score, which corresponds to an unweighted average of the MUC, B^3 and $CEAF_e$ F1 scores. Micro-averaging is used throughout when reporting our scores for entire dataset CoNLL.

Settings All our perceptron models are trained on 30 iterations on the corpus, using the tree loss defined above and max-loss update in structured mode.

6.5.2 Results and discussion

Our main results are summarized in Table 6.1.

	MUC			B^3			$CEAF_e$			CoNLL
	P	R	F1	P	R	F1	P	R	F1	
Baselines										
CLOSESTFIRST	88.0	75.6	81.3	81.0	63.3	71.0	51.0	74.8	60.6	71.0
TRANSITIVE	87.1	82.9	84.9	66.1	77.4	71.3	53.7	62.4	57.7	71.3
BESTFIRST	89.7	77.0	82.9	84.2	66.0	74.0	52.0	76.7	62.0	73.0
Structured Models										
MSF^{struct}	89.1	77.8	83.1	76.5	68.6	72.3	49.7	70.7	58.3	71.3
BESTFIRST ^{struct}	85.4	85.0	85.2	69.9	75.2	72.4	67.1	68.1	67.6	75.1
Constrained Models										
MSF^{constr}	90.6	79.5	84.7	81.4	70.8	**75.7**	51.8	73.0	60.6	73.7
BESTFIRST ^{constr}	86.4	85.4	**85.9**	73.9	76.3	75.1	67.3	69.9	**68.6**	**76.5**

Table 6.1: Results on CoNLL-2012 Test Set (gold mentions).

Local vs. global First, we compare our three local models (the baselines) with structured versions of BESTFIRST and MSF. As expected, CLOSESTFIRST and BESTFIRST tend to have larger precision scores and higher recall scores on MUC and B^3 (and the other way around for $CEAF_e$) compared to transitive closure, which relates their propension for smaller (resp. larger) clusters. What is remarkable in the structured version is that overall, BESTFIRST is significantly improved, and this mostly comes from $CEAF_e$ improvement, which indicates that the entities are better "segmented" in the sense that they align better with the *gold* partition. Compared to its local equivalent, the structured BESTFIRST changes the balance for more recall. By contrast, the CoNLL score of MSF does not improve over its local counterpart

(although the structured MSF is apparently more precise). The superiority of BESTFIRST over MSF is likely to come from the fact that it is linguistically more realistic as humans process mentions sequentially. BESTFIRST structures may also be more advantageous from a learning point of view in that they define a smaller, more biased space; they also probably give rise to fewer model updates.

Unconstrained vs. constrained When considering models learned with (or without) constraints, let us first note that MSF^{constr} is a new model and does not appear in previous work. The results are very clear here: constraints improve performance across all metrics, both in precision and recall, providing an improvement of one and a half CoNLL point. The constrained version of the structured BESTFIRST has a CoNLL score of 76.5 on *gold mentions*, which would place us second in CoNLL-2012 Shared Task in the same setting. This result plays in favor of mixed coreference models, with both feature description of pairs and deterministic constraints.

Sparse vs. dense The next set of experiments explicitly compares the performance of sparse and dense latent structure learning. So, it is only related to "global" models. We only study MSF transitive closure since BESTFIRST does not have obvious dense dual structure. MSF learning is the latent structure as calculated in section 6.3. In TRANSITIVE^{struct} model, the cluster are represented by the sum of the link of the corresponding cliques. The third model, $G^{+struct}$, is intermediate: it uses the graph G^{+} as latent structure, and the "true value" employed to update the model is given by the cliques (transitive closure). Results are given in table 6.2. What we observe is that the dense structure hamper the model's results. The worse configuration is when using all links in TRANSITIVE^{struct}. Basically, what happens is that dense structured models tend to have more aggressive and inaccurate updates. This experiment tells us that sparse latent structures are better suited to learning. We saw in table 6.1 that they are also adapted to constrained learning.

	MUC			B^3			$CEAF_e$			CoNLL
	P	R	F	P	R	F	P	R	F	
MSF^{struct}	89.1	77.8	**83.1**	76.5	68.6	**72.3**	49.7	70.7	**58.3**	**71.3**
TRANSITIVE^{struct}	88.7	64.9	75.0	84.7	56.6	67.9	37.1	70.2	48.5	63.8
$G^{+struct}$	85.3	71.8	78.0	73.8	63.5	68.3	41.8	64.0	50.6	65.6

Table 6.2: Sparse vs dense structure learning on CoNLL-2012 Test Set (gold mentions).

Densifying decoders So far, we have shown that BESTFIRST empirically outperforms MSF for learning latent structures, but there exists a whole family of decoders whose expressive power is between that of BESTFIRST and that of MSF. Our final set of experiments, carried out on the CoNLL-2012 Dev set, studies these other decoders. Specifically, we start with a

pairwise model learning on the Train Set and work on the G^+ we obtain with it. We define intermediate parameterized decoding strategies that allow us to *densify* the set of selected links compared to initial decoders. In order to perform exhaustive comparisons, we only run each new decoder with a local mention pair model (i.e., the model used as baseline in section6.5).

First, we *densify* BESTFIRST[6] by considering the n best backward positive link selection (it can be less than n for a given mention if we do not have enough positive backward links at our disposal). This strategy obviously converges to the transitive closure. The effect of increasing n is illustrated in B^3 metric on figure 6.4: as we can imagine, precision decreases and recall increases, but overall, the F1-score decreases as soon as we add supplementary backward positive links.

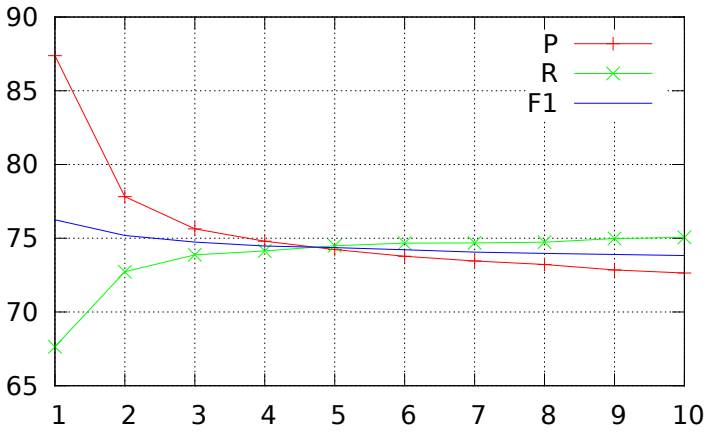

Figure 6.4: n-BESTFIRST B^3 convergence (Dev Set)

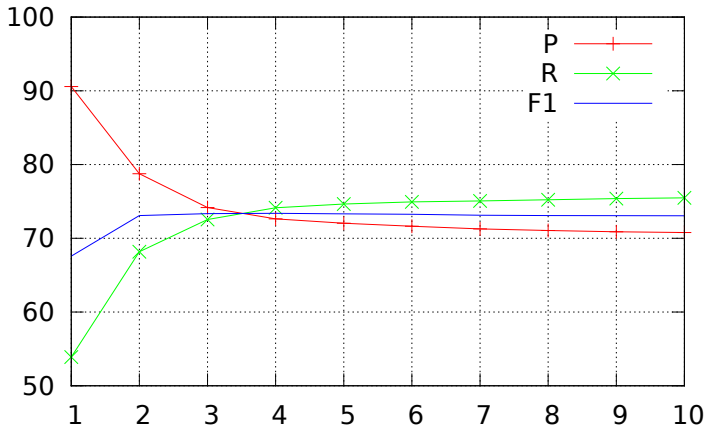

Figure 6.5: n-BESTLINK B^3 convergence (Dev Set)

We next define a second densifying strategy, n-BESTLINK, selecting for each mention the best n links to any mention (irrespective of their relative position). This is a sort of generalization of BESTFIRST. As shown in figure 6.5, this model starts with a higher precision

[6]No similar generalization of CLOSESTFIRST, and MSF is already a transitive closure and cannot be densified further.

than n-BESTFIRST, but surprisingly, contrarily to n-BESTFIRST, the B^3 score augments as the strategy densifies.

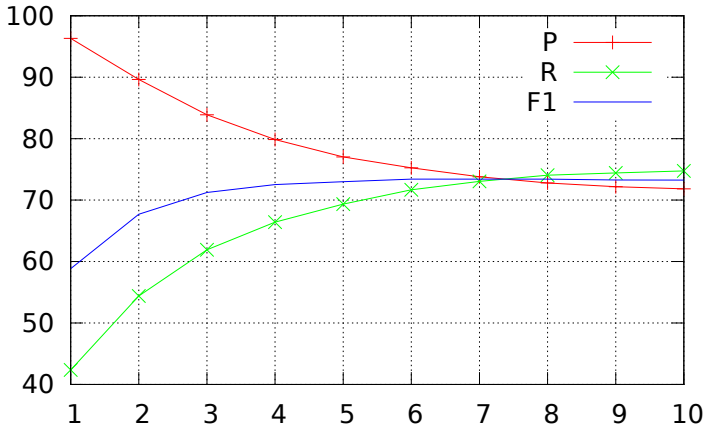

Figure 6.6: n-BESTEDGE B^3 convergence (Dev Set)

Finally, we define a model n-BESTEDGES that selects the n best edges of G^+. This is a sort of relaxation of Kruskal algorithm, which may create cycle at each link selection. Because the size of document varies, we do not actually take the n best links, but a $n \times$ *a function depending on the size of the document*. To make it simple, we fit a power law for each category of documents on the Train Set giving the approximate number of coreference links in function of the number of mentions. We then select n tenth of this number best edges to get a smooth evolution. Here, we start with a very high precision, and the F1 increase as we select more links (see figure 6.6).

To summarize, we provide the corresponding CoNLL for each of the three strategies in Figure 6.7. Let us make it clear that the rate of convergence cannot be compared between the strategies. Instead, we look at the general behavior. Obviously, densifying the strategies make them converge to the transitive closure. But what is striking is that only n-BESTFIRST is hindered by densifying (in fact it is optimal with $n = 1$) while the two other similar strategies are improved by being more dense. A possible explanation for that is that BESTFIRST is related to the linguistic intuition that resolving coreference can be achieved through finding an antecedent to each anaphora. BESTFIRST is optimal in the sense that it is the sparsest structure obtaining the best results, however, it is not the best we can do given the positively classified links: we computed an oracle giving the best clustering we can get from it and obtained a CoNLL score of 79.86.

6.6 Related work to this chapter

For a more detailled presentation of the state of the art in coreference resolution, look back to chapter 4. This section gives a few description of work related to this chapter.

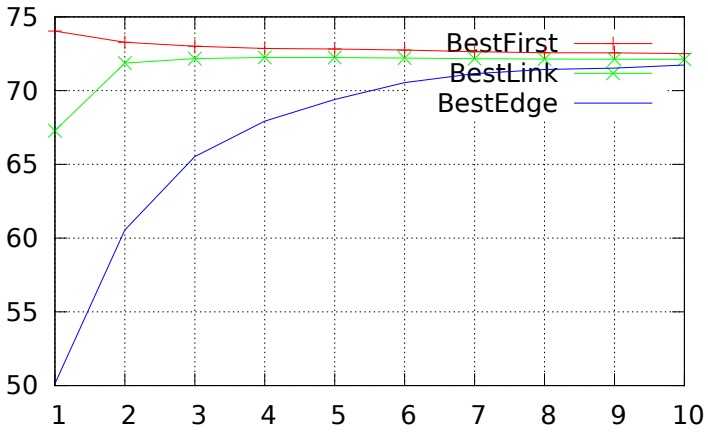

Figure 6.7: n-Best CoNLL score convergence (Dev Set)

A popular supervised learning approach to coreference resolution is the mention-pair model (McCarthy & Lehnert, 1995; Morton, 2000; Soon *et al.*, 2001; Ng & Cardie, 2002b; Stoyanov *et al.*, 2010a; Lassalle & Denis, 2013b; Björkelund & Farkas, 2012) which, despite its simplicity, works well in practice. However, a drawback of this approach is that the classifier is optimized independently from the clustering. Recent structured output models directly address this problem. While it is difficult to globally optimize a clustering metric (it can be a NP-hard problem), one can instead optimize over the latent tree structures that represent the most informative links in the clusters (Fernandes *et al.*, 2012; Chang *et al.*, 2013; Yu & Joachims, 2009).

Other structured output models to coreference include correlation clustering (Finley & Joachims, 2005) as well as various probabilistic graphical model-based approaches (McCallum & Wellner, 2004; Culotta *et al.*, 2007). These learning models are more complex in that they also attempt to enforce transitivity. Other transitivity enforcing models use Integer Programming-based (Klenner, 2007; Denis & Baldridge, 2009). Due to their much higher complexity, these global decoding schemes are used in combination with locally-trained models. Coreference resolution has also been framed as a (hyper)graph-cut problem (Nicolae & Nicolae, 2006; Cai & Strube, 2010a). Several other models have attempted to break away from the mention pair representation altogether, trying to model cluster-mention or cluster-cluster relations (Luo *et al.*, 2004; Haghighi & Klein, 2010; Rahman & Ng, 2011; Stoyanov & Eisner, 2012).

6.7 Conclusion and outlooks

We thoroughly studied the structural properties of common decoders used for learning-based coreference resolution and saw how they were related to transitive closure. In particular, we showed that BESTFIRST decoding produces subpartitions of transitive closure and that empirically, this sparse link selection outperforms other strategies very well both with locally-trained

and structured models. In the context of structured learning, we saw that sparse latent structures were much more suited to learning than dense structures, and that well-chosen constraints allow for additional performance gains. By comparing models into the same framework, we can definitely say that BESTFIRST structure is more suited to coreference resolution than MSF: even if the latter explores a very large space, it does not produce better clustering. Finally, finding accurate must-link and cannot-link constraints from various sources remains an open area, and we hope they can improve structured models.

Chapter 7

Joint Anaphoricity Detection and Coreference Resolution by Learning Constrained Latent Structures

The results of this chapter have been published as a long paper in AAAI 2015 (Lassalle & Denis, 2015).

So far, we have defined a technique to improve linear models that are usually employed in coreference resolution (see Chapter 5), and we carried out experiments on various treelike structures and decoders to find out that sparse latent structures were suited to learning to resolve coreference in a structured (global) way. The BESTFIRST *tree appeared to be the best combination for coreference resolution (see Chapter 6). In this chapter, we address our third objective (see chapter 1), that is to say, we extend the treelike structures to resolve coreference jointly with another task: anaphoricity. We aim at improving the quality of coreference clusters by replacing a pipeline architecture, where errors propagate, by a joint structure, where all information is processed at the same time. Because we employ the same kind of structures, we are still addressing coreference resolution globally, i.e., at the document level, and thus we are defining a model which is both global and joint.*

This chapter introduces a new structured model for learning anaphoricity detection and coreference resolution in a joint fashion. Specifically, we use a latent tree to represent the full coreference and anaphoric structure of a document at a global level, and we jointly learn the parameters of the two models using a version of the structured perceptron algorithm. Our joint structured model is further refined by the use of pairwise constraints which help the model to accurately capture certain patterns of coreference. Our experiments on the CoNLL-2012 English datasets show large improvements in both coreference resolution and anaphoricity detection, compared to various competing architectures. Our best coreference system obtains a CoNLL score of 81.97 *on gold mentions, which is to date the best score reported on this setting of the dataset.*

7.1 Introduction

Resolving coreference in a text, that is, partitioning *mentions* (noun phrases, verbs, etc) into referential *entities* is a challenging task in NLP leading to many different approaches (Ng,

2010). Anaphoricity detection, on the other hand, consists in deciding whether a mention is *anaphoric* (aka *discourse-old*) or *non-anaphoric* (*discourse-new*).[1] This task is strongly related to coreference resolution and has been mainly addressed as a preliminary task to solve, leading to pipeline architectures (Ng & Cardie, 2002a; Ng, 2004b; Denis & Baldridge, 2008) (see Chapter 8 for a description of the pipeline).

An important drawback of pipelined models is that errors tend to propagate from anaphoricity detection to coreference resolution, hence ultimately hindering the performance of the downstream system. In order to avoid error propagation, Denis & Baldridge (2007) propose a joint inference scheme using Integrer Linear Programming (ILP) to maximize the scores of both of the two models. In this case, inference is performed jointly but the two models are still trained independently. Poon & Domingos (2008) perform joint learning using using Markov Logic Networks, but sampling techniques are needed to perform inference. Rahman & Ng (2011) propose a ranking approach wherein, for each mention taken in the order of the text, the decision to link it to a previous mention and to classify it as discourse-new is taken jointly. In this approach, the decision is local to the mention and the previous context, but crucially does not take into account the next mentions in the document. Other approaches simply use the output of an anaphoricity classifier as feature for the coreference model (Bengtson & Roth, 2008; Durrett *et al.*, 2013).

In this chapter, we employ latent trees to represent the full coreference and anaphoricity structure of a document. We extend latent tree models that can be found in Yu & Joachims (2009); Fernandes *et al.* (2012); Chang *et al.* (2013); Björkelund & Kuhn (2014) by introducing two kinds of edges: the first ones encode coreference links, while the second ones represent *discourse-new* elements. Basically, a latent coreferent tree links together the mentions that make up the same entity. We restrict the shape of latent trees by allowing only one “backward link" per mention so as to be able to define a coherent structure when introducing discourse-new links. This also allows us to compute the structure easily from a weighted graph using a greedy “Best-First" algorithm. Our main contribution is to provide the first system that learns coreference resolution and anaphoricity detection both in a *joint* and *global* fashion. The model is joint in that parameters for the two models are estimated together, so that changes in the anaphoricity detection model directly affect the estimation of the coreference resolution parameter (and vice versa). The model is global in that parameters are learned in a way that minimizes a loss defined at the document level. We additionally define a set of must-link and cannot-link constraints on the structure, which helps improving the model on certain types of coreference links such as coreference of first person pronouns with the speaker of a sentence.

Our experiments on CoNLL-2012 compare pipeline vs. joint models and local vs. global versions of them, always obtaining better coreference results with joint models. More precisely, the CoNLL score systematically improves as one goes from pipeline to joint models as well as from local to global models. The constrained version of our global joint model obtains the best

[1] In this chapter, we slightly overload these terms by taking a non-anaphor to denote the first mention of an entity (in the order of the text), and an anaphor any mention that is not.

results overall, and achieves performance that is well above the state-of-the-art. At the same time, we observe that anaphoricity detection also largely improves in the global joint model.

The rest of the chapter is organized as follows. In Section 7.2, we define our latent tree structures and explain why we limit the tree to certain shapes. We then detail the learning procedure in Section 7.3, the features of our models in Section 7.4, and our results are finally presented in Section 7.5.

7.2 Joint Latent Representation of the Coreference Structure

This section first discusses the relationship between anaphoricity and coreference, and then defines a tree structure to represent the coreference structure of a document. We explain how one can restrict the shape of latent trees to compute them efficiently.

7.2.1 Anaphoricity and Coreference

Once mentions are identified in a text, they have to be linked together to form coreference clusters. A question one may ask is whether a mention is *anaphoric*, that is, is understood through a reference to one or several preceding mentions. If not, it is a *discourse-new*. Determining anaphoricity helps us reducing the search space for coreference resolution. For example, in the text in figure 7.1, m_1, m_2 and m_5 are discourse-new (i.e., they do not have backward coreference link to preceding mentions), while all other mentions are anaphora (i.e., they have outgoing backward coreference links).

[Peter]$_{m_1}$ told **[John]**$_{m_2}$ **[he]**$_{m_3}$ did not manage to open the door of **[his]**$_{m_4}$ apartment because **[the key]**$_{m_5}$ broke off in the lock. **[[His]**$_{m_6}$ **friend]**$_{m_7}$ managed to get **[it]**$_{m_8}$ out and open, which made **[him]**$_{m_9}$ very happy.

Figure 7.1: A simple example: only mentions with coreference links (i.e., non-singleton) are annotated.

A common approach is to detect anaphoricity before coreference resolution, giving rise to a pipeline architecture: mentions are classified as anaphoric or not based on a locally trained model, and these classifications are used to constrain the decisions of the downstream coreference model. That is, only mentions that are classified as anaphoric can be linked to a previous mention by the coreference classifier. An important drawback of such systems is that errors tend to propagate, which in turn requires a careful tuning of the confidence threshold used in anaphoricity classification Ng (2004b).

7.2.2 Joint Representation of Anaphoricity and Coreference

Some coreference systems implement a joint determination of anaphoricity and resolution coreference. Rahman & Ng (2011) propose a ranking approach which jointly learns the two tasks but not at the document level (the decision is taken regardless of the coreference structure to be built after). Denis & Baldridge (2007) formulate the joint problem as an ILP problem that globally maximizes the scores of an anaphoricity model and a pairwise coreference model subject to constraints relating the two tasks (e.g., resolve all and only anaphors). However, this approach has two disadvantages: the complexity of the ILP problem to solve (i.e., it is NP-hard), and more importantly the fact that the models are trained separately (and thus do not incorporate global decisions).

Latent tree representations of coreference clusters have proven efficient for globally learning to resolve coreference (Fernandes *et al.*, 2012; Chang *et al.*, 2013; Yu & Joachims, 2009; Björkelund & Kuhn, 2014). We start from an undirected weighted graph of pair mentions (see section 7.3 for details about weighting the graph) and a collection of trees is computed, each tree representing a cluster. Two methods have been used for building such trees from weighted graphs: running a Maximum Spanning Tree (MST) algorithm on the graph or using a BESTFIRST decoding strategy (i.e., only the highest scoring backward link is selected for each mention, provided it exists). It is easy to see that the set of links selected by this latter method has no cycle, and then is also a spanning forest (Björkelund & Kuhn, 2014). That is, both methods yield spanning forests but the structures generated by the latter method have a more restricted topology. From now on, we will refer to this second method as BESTFIRST MST.

There are at least two main motivations for using BESTFIRST MST over standard MST. First, the experiments carried out by Chang *et al.* (2013) suggest that BESTFIRST trees achieve better results than MST on coreference resolution. Second, BESTFIRST MST appear to be easy to calculate from an algorithmic point of view. Thus, the BESTFIRST strategy can be easily extented by defining a single *rooted* tree for representing the partition of mentions. The root is a dummy mention added to the other mentions and placed before them in the order of the text. Root-mention links directly encode the fact that the mention is a discourse-new, while mention-mention links are coreference links (see figure 7.2). This interpretation of root-mention links is guaranteed by BESTFIRST MST because no coreference path can be created between a mention linked to the root and a previous mention. We give a sketch of proof for this result: if such a path existed, the rightmost element (in the order of the text) of the set of mentions occurring along the path would necessarily have two backward links, which is not possible with the BESTFIRST strategy. By contrast, this kind of path is allowed in unrestricted MST: imagine for instance that we have two mentions m_1 and m_2 and that the MST contains the links $(root, m_2)$ and (m_1, m_2). We see that the semantics of "root-mention" links is not preserved in that case.

Formally, for a given document with mentions $\{m_1, \ldots, m_n\}$, we consider a complete

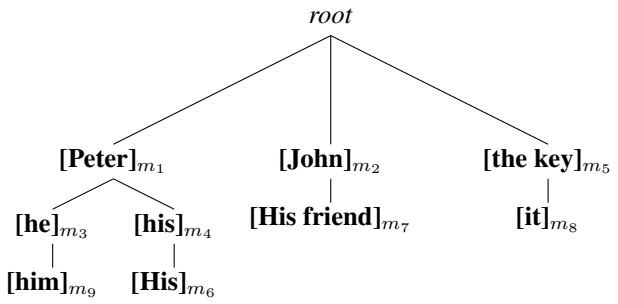

Figure 7.2: A latent tree representing the coreference structure of the text in figure 7.1.

weighted undirected graph $G = (V, E, \omega)$, where V is the set of mentions plus an additional *root* (ordered as $root \preceq m_1 \preceq \cdots \preceq m_n$), E all the pairs formed using V, $\omega : E \rightarrow \mathbb{R}$ a weighting function decomposed as follows:

$$\begin{cases} \omega(m_i, m_j) = \omega_c(m_i, m_j) & 1 \leq i, j \leq n \\ \omega(root, m_j) = \omega_n(m_j) & 1 \leq j \leq n \end{cases}$$

where $\omega_c : E \rightarrow \mathbb{R}$ quantifies the confidence that a pair is coreferent, and $\omega_n : V \rightarrow \mathbb{R}$ the confidence that a mention is discourse-new. We explain how to learn this weighting function in section 7.3. We define $\mathcal{S}^{best}(G)$ as the set of spanning trees on G such that every mention can only have at most one link to a previous mention (or to the *root*). We want to compute the following structure:

$$\hat{T} = \underset{T \in \mathcal{S}^{best}(G)}{\arg\max} \sum_{e \in E_T} \omega(e)$$

with E_T the set of links of the spanning tree T. It is easy to see why it is a global decision, that jointly incorporates coreference and anaphoricity, by decomposing the objective function as:

$$\sum_{(m,m') \in E_T^{coref}} \omega_c(m, m') + \sum_{m \in V_T^{new}} \omega_n(m)$$

where E_T^{coref} is the set of links (m_i, m_j) in tree T such that $m_i \neq root$ and V_T^{new} the set of mentions m_j such that there is a link $(root, m_j)$ in the tree.

Because we have restricted the shape of spanning trees, we can compute the *argmax* easily by using a BESTFIRST strategy (see figure 7.3): for each mention, the backward edge with the highest weight is selected, and links the mention either to a previous mention (i.e., it is anaphoric) or to the root (i.e., it is discourse-new).

From a topological point of view, our tree is similar to the one used by Fernandes *et al.* (2012). The difference is that they do not have weights on root-mention links (no global anaphoricity detection), and they compute the structure with Chu-Liu-Edmonds Algorithm (Chu & Liu, 1965; Edmonds, 1965). However, as pointed out by Björkelund & Kuhn (2014), because MST is computed on an oriented graph with only backward edges, it is sufficient to use a BESTFIRST strategy to build it.

INPUT: $G = (V, E, \omega)$
OUTPUT: BESTFIRST Spanning Tree T

$E_T = \varnothing$
for $j : 1 \dots N$ **do**
  $m_{best} = \varnothing$; $s_{best} = -\infty$
  for $i : 1 \dots j-1$ **do**
    if $\omega_c(m_i, m_j) > s_{best}$ **then**
      $m_{best} = m_i$; $s_{best} = \omega_c(m_i, m_j)$
    end if
  end for
  if $\omega_n(m_j) > s_{best}$ **then**
    $m_{best} = root$
  end if
  add link (m_{best}, m_j) to E_T
end for

Figure 7.3: Computing the coreference structure with the highest weight.

7.2.3 Constrained Structures

A way to integrate prior knowledge of the coreference structure is to use constraints on mention pairs: we can add *must-link* (knowledge of a coreference link) and *cannot-link* (impossibility of linking two mentions) constraints in the computation of the spanning tree. These constraints can be generated by finding patterns in the text using accurate rules (see section 7.4 for details). In this case, the BESTFIRST strategy only creates a backward link for mentions that do not appear at the right position of a must-link, and backward links are selected among those which are not cannot-links.

7.3 Learning Latent Structures

In this section, we explain how ω_c and ω_n are learned from data. We formulate the problem of learning coreference partitioning and anaphoricity dectection as a joint structured classification problem, following previous recent approaches (Yu & Joachims, 2009; Fernandes *et al.*, 2012; Chang *et al.*, 2013; Björkelund & Kuhn, 2014).

7.3.1 Structured Learning

In this formulation, the learning algorithm observes a set of examples (i.e., annotated documents of the training set) $\mathcal{T} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{T}$, where $\boldsymbol{x}_i$ is an instance from a structured input space $\mathcal{X}$ (the space of documents) and $\boldsymbol{y}_i$ is a *structured* label from an output space $\mathcal{Y}$ whose size is typically exponential in the size of $\boldsymbol{x}_i$. Suppose we have at hand a *joint feature map* $\Phi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$. We additionally assume that $\mathcal{H} = \mathcal{H}_c \oplus \mathcal{H}_n$ and that the subspaces are equipped with their own inner products $\langle \cdot, \cdot \rangle_c$ and

$\langle \cdot, \cdot \rangle_n$ induced from $\langle \cdot, \cdot \rangle$. Now, given a document $\boldsymbol{x}$ with n mentions $m_1, \ldots, m_n$, we create a graph G with additional *root* as described in Section 7.2. Let $\boldsymbol{y}$ be a BESTFIRST MST on G with coreference links E_y^{coref} and anaphoric mentions V_y^{new}. We restrict Φ to the following decomposed form:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(m,m') \in E_y^{coref}} \phi_c(\boldsymbol{x}, m, m') + \sum_{m \in V_y^{new}} \phi_n(\boldsymbol{x}, m)$$

where $\phi_c(\boldsymbol{x}, m, m')$ (resp. $\phi_n(m)$) is an embedding of the mention pair (m, m') (resp. of the mentions m) in $\mathcal{H}_c$ (resp. $\mathcal{H}_n$), depending on information contained in document $\boldsymbol{x}$ (e.g. a feature representation). To score the pair $(\boldsymbol{x}, \boldsymbol{y})$, we suppose we have at hand two weight vectors $\boldsymbol{w}_c \in \mathcal{H}_c$ and $\boldsymbol{w}_n \in \mathcal{H}_n$. The score associated to $(\boldsymbol{x}, \boldsymbol{y})$ is:

$$\begin{aligned} \langle \boldsymbol{w}, \Phi(\boldsymbol{x}, \boldsymbol{y}) \rangle = & \sum_{(m,m') \in E_y^{coref}} \langle \boldsymbol{w}_c, \phi_c(\boldsymbol{x}, m, m') \rangle_c \\ & + \sum_{m \in V_y^{new}} \langle \boldsymbol{w}_n, \phi_n(\boldsymbol{x}, m) \rangle_n \end{aligned}$$

The relationship with the weighting function on G described in Section 7.2 is the following: $\omega_c(m, m') = \langle \boldsymbol{w}_c, \phi_c(\boldsymbol{x}, m, m') \rangle_c$ and $\omega_n(m) = \langle \boldsymbol{w}_n, \phi_n(\boldsymbol{x}, m) \rangle_n$. Predicting $\boldsymbol{x}$'s BESTFIRST MST amounts to solving:

$$T(\boldsymbol{x}) = \operatorname*{arg\,max}_{T \in \mathcal{S}^{best}(G)} \langle \boldsymbol{w}, \Phi(\boldsymbol{x}, \boldsymbol{y}) \rangle$$

Now, we must address the problem of learning relevant weights vectors $\boldsymbol{w}_c$ and $\boldsymbol{w}_n$. The problem is that we cannot directly observe *gold* BESTFIRST MST since we are only given annotation describing the coreference partition of documents, and that there is no unique tree corresponding to a given partition. Hence, we have to find a way to create targeted structures from the clustering information. In the following, we address the problem in the structured perceptron framework.

7.3.2 Latent Structure Perceptron-based Learning

For our purpose, we consider that $\mathcal{H}_c = \mathbb{R}^m$ and $\mathcal{H}_n = \mathbb{R}^p$ and we use the canonical dot products. The goal of learning is to acquire the weight vectors $\boldsymbol{w}_c \in \mathbb{R}^m$ and $\boldsymbol{w}_n \in \mathbb{R}^p$. We estimate these parameters with the online perceptron algorithm presented in Figure 7.4.

This algorithm is only given a sequence $\{\boldsymbol{x}_i, P_i\}_{i=1}^T$ of documents without their corresponding spanning trees, but with information on the partition of their mentions. Most of the time there are several possibilities for representing a partition by a tree, and we need to select one at each round. Starting from initial weight vectors $\boldsymbol{w}_c^{(0)}$ and $\boldsymbol{w}_n^{(0)}$, it iterates N times over the training examples, giving a total of $N \times T$ iterations. At each round i, a *true* tree is

INPUT: Training data: $\mathcal{T} = \{(\boldsymbol{x}_i, P_i\}_{i=1}^{T}$
OUTPUT: Weight vectors $\boldsymbol{w}_c$ and $\boldsymbol{w}_n$

$\boldsymbol{w}_c^{(0)} = 0; \boldsymbol{w}_n^{(0)} = 0; \boldsymbol{v}_c = 0; \boldsymbol{v}_n = 0; i = 0$
for $n : 1..N$ **do**
  for $t : 1..T$ **do**
    Compute *true* label $y_i^{\boldsymbol{w}}$ from P_i and $(\boldsymbol{w}_c^{(i)}, \boldsymbol{w}_n^{(i)})$:

$$y_i^{\boldsymbol{w}} = \arg\max_{y \in \tilde{\mathcal{Y}}_i} \Big\{ \sum_{(m,m') \in E_y^{coref}} \langle \boldsymbol{w}_c, \phi_c(\boldsymbol{x}, m, m') \rangle_c + \sum_{m \in V_y^{new}} \langle \boldsymbol{w}_n, \phi_n(\boldsymbol{x}, m) \rangle_n \Big\}$$

    Compute *max-loss* prediction $\tilde{y}$:

$$\tilde{y} = \arg\max_{y \in \mathcal{Y}} \Big\{ \sum_{(m,m') \in E_y^{coref}} \langle \boldsymbol{w}_c, \phi_c(\boldsymbol{x}, m, m') \rangle_c + \sum_{m \in V_y^{new}} \langle \boldsymbol{w}_n, \phi_n(\boldsymbol{x}, m) \rangle_n + l(y, y_i^{\boldsymbol{w}}) \Big\}$$

$$\boldsymbol{w}_c^{(i+1)} = \boldsymbol{w}_c^{(i)} + \sum_{(m,m') \in E_{y_i^{\boldsymbol{w}}}^{coref}} \phi_c(\boldsymbol{x}, m, m') - \sum_{(m,m') \in E_{\tilde{y}}^{coref}} \phi_c(\boldsymbol{x}, m, m')$$

$$\boldsymbol{w}_n^{(i+1)} = \boldsymbol{w}_n^{(i)} + \sum_{m \in V_{y_i^{\boldsymbol{w}}}^{new}} \phi_n(\boldsymbol{x}, m) - \sum_{m \in V_{\tilde{y}}^{new}} \phi_n(\boldsymbol{x}, m)$$

    $\boldsymbol{v}_c = \boldsymbol{v}_c + \boldsymbol{w}_c^{(i+1)}$
    $\boldsymbol{v}_n = \boldsymbol{v}_n + \boldsymbol{w}_n^{(i+1)}$
    $i = i + 1$
  end for
end for
$\boldsymbol{w}_c = \frac{\boldsymbol{v}_c}{(N \times T)}; \boldsymbol{w}_n = \frac{\boldsymbol{v}_n}{(N \times T)}$

Figure 7.4: Structured perceptron learning algorithm with parameter averaging, in *max-loss* mode for joint learning of coreference resolution and anaphoricity detection.

computed for document $\boldsymbol{x}_i$, and it plays the role of the *gold* label in structured learning. We select the tree $y_i^{\boldsymbol{w}}$ with the best current weight which is compatible with the partition P_i (in the algorithm, the set of those trees is denoted by $\tilde{\mathcal{Y}}_i$. This *gold tree* is easily computed by removing non-coreferent edges from the complete graph according to the *gold* clustering, and by applying the extended BESTFIRST strategy described in figure 7.3.

Next, we compute a predicted structure in *max-loss* mode (Crammer *et al.*, 2006).[2] This corresponds to a trade-off between maximizing the weight of the predicted tree and finding a tree "far from the *true* tree", according to a loss counting the number of edges the structures do not have in common (the approach is similar to (Fernandes *et al.*, 2012)). Again the predicted tree is computed by modifying the weights of the graph (by just adding one to all links not in the *true* tree), and applying the extended BESTFIRST. The weight vectors are updated by the difference of *true* and *max-loss* predicted labels in a structured perceptron manner (the difference is projected in $\mathcal{H}_c$ and $\mathcal{H}_n$ to update $\boldsymbol{w}_c$ and $\boldsymbol{w}_n$ respectively). The final weight vector is obtained by averaging over the weight vectors compiled after each round. Weight averaging is common in online learning for it helps reducing overfitting Freund & Schapire (1999); Collins (2002).

[2] The more direct *prediction-based* (i.e., without loss) update always gave lower results in our development experiments, so we do not detail this learning mode here.

Now we see that coreference resolution and anaphoricity detection are learned both in a *jointly* and *globally* manner: jointly because backward coreference links are in balance with "anaphoricity links", and globally beacause the update is achieved on a tree representing the complete coreference structure of the document.

7.3.3 Constrained Learning

As explained in Section 7.2, simple *must-links* and *cannot-links* rules can be applied before using the learning model. Specifically, we remove cannot-links from the graph and add obligatory edges to the tree and complete the tree by applying the BESTFIRST rule on mentions that are not at the right hand position of a must-link and by avoiding removed links. This is done both during training and inference.

7.4 Systems Description

7.4.1 Local vs. Structured models

Our different coreference systems are based on pairwise representation of mention pairs, meaning we re-employ standard pairwise features. We define a *baseline* system, referred to as "local model" which is a simple pairwise model trained using an averaged perceptron, and using a BESTFIRST decoder (Ng & Cardie, 2002b; Bengtson & Roth, 2008). This model uses anaphoricity information in the form of a feature that corresponds to the output of an anaphoricity mention classifier in a way similar to Bengtson & Roth (2008) and Durrett *et al.* (2013). This model is also trained using the averaged perceptron.

We also define a joint local model JOINTBESTFIRST whose behavior is the same as BESTFIRST with the difference that the root can be an antecedent (using two separates weight vectors to represent coreference and anaphoricity). It is joint in that the decision of creating a backward link is opposed to the decision of classifying the mention as discourse-new. But it is still local because the model is updated for each mention of a document.

As opposed to these local models, we set up a "global model", which consists in learning BESTFIRST MST structures with the algorithm given in Section 7.3, JOINTBESTFIRST^{struct}. We compare this model to its version without anaphoricity BESTFIRST^{struct} (global coreference, but not joint). For the two global models, we also define additional constrained versions, JOINTBESTFIRST^{constr} and BESTFIRST^{constr}.

7.4.2 Pipeline vs. Joint models

We compare joint models to their pipeline equivalents by using a classifier of anaphoricity upstream. PIPEBESTFIRST is the pipelined version of BESTFIRST: that is, backward links are forbidden for mentions detected as discourse-new. Conversely, a mention that is classified as anaphoric by the anaphoricity model must have a backward link (taking the one with the highest score, even if negative). Similarly, we define pipeline versions of the structured models, PIPEBESTFIRST^{struct} and PIPEBESTFIRST^{constr}.

7.4.3 Feature sets

Coreference Features Our system uses a classical set of features used for mention pair classification (for more details see Bengtson & Roth (2008); Rahman & Ng (2011). These include: grammatical types and subtypes, string and substring match, apposition and copula, distance (number of separating mentions/sentences/words), gender and number match, synonymy/hypernymy and animacy, family name (based on lists), named entity types, syntactic features (gold parse), a morphological feature indicating if a verb is derived from a noun and anaphoricity detection. In addition, we use products of the above features with grammatical types, which we found to improve the results of all our models.

Anaphoricity Features For the anaphoricity classifier, we also re-use features proposed in previous work (Ng & Cardie, 2002a; Ng, 2004b; Denis & Baldridge, 2008). These include: number of words in the mention; binary features indicating if it is pronoun, speech pronoun, reflexive pronoun, proper name, definite description, quantified description, possessive description or bare noun; the position in text; if the mention is embedded in another mention; if the string/the head matches that of a preceding mention; if the mention is an apposition or acronym of a preceding mention.

7.4.4 Constraints

We defined a rather small set of constraints, but sufficient to improve the performance of our models. Our *must-links* are given, first, by our own implementation of sieve 1 of Lee *et al.* (2011), which accurately matches patterns involving the speaker of sentences (e.g. *He* said: "*I* believe you"), and second, by exact string matches of proper nouns. We also use several sets of *cannot-links*, coming from number, gender, and (un)animated mismatches, as well as i-within-i constraints.

In addition, in all our models (constrained or not), we systematically set cannot-links between pronouns to make the BESTFIRST approach coherent with Ng & Cardie (2002b) (no pronoun antecedent for pronouns).

7.5 Experiments

In these experiments, our objective is to evaluate the improvement one can get with a joint structured model over their pipelined and/or local counterparts, both from coreference resolution and anaphoricity detection perspectives.

7.5.1 Experimental setup

Data Our models are evaluated on the CoNLL-2012 Shared Task English corpus. We use the official separation of documents in Train/Dev/Test data sets and we test our models in the *closed mode* in which features can be built only from provided data (with the exception of two additional sources: WordNet and Bergsma & Lin (2006)'s gender and number data).

We evaluate our models on the *gold mentions*, to avoid introducing noise from detected mention and focus more on anaphoricity detection and coreference resolution. A full end-to-end coreference resolution system might require additional work and heuristics to filter out some mentions in order to build a robust coreference system.

Evaluation Our models are evaluated using the following three popular coreference resolution metrics: MUC (Vilain *et al.*, 1995), B^3 (Bagga & Baldwin, 1998a), and Entity-based CEAF (or $CEAF_e$) (Luo, 2005). Following Pradhan *et al.* (2012), we also report a global F1-score, refered to as the CoNLL score, which corresponds to an unweighted average of the MUC, B^3 and $CEAF_e$ F1 scores. Micro-averaging is used throughout when reporting our scores for entire CoNLL-2012 dataset.

Settings All our models are (local or structured) linear models, learned with the averaged perceptron algorithm with 30 iterations on the corpus (it is sufficient to obtain stable scores on the Dev set). In structured learning, we used the max-loss learning mode, associated with the tree loss defined in Section 7.3. Our baseline for anaphoricity detection is a simple averaged perceptron.

7.5.2 Results and Discussion

Coreference resolution All the coreference results are reported in table 7.1. We observe the following improvements: whether pipeline or joint, structured models perform better than local models and constrained models better than unconstrained models.

Notice that the local pipelined model PIPEBESTFIRST slightly improves over the local BESTFIRST (from 72.96 to 73.46), but its structured version PIPEBESTFIRST^{struct} performs

a little worse than BESTFIRST^{struct} (from 75.07 down to 74.4), mostly due to precision losses on MUC and B^3 and a corresponding loss in recall on $CEAF_e$. It is unclear whether these differences are truly significant, but this might mean that deciding whether to use the pipeline as a hard constraint or to only propagate anaphoricity values through a feature depends on the coreference model we have chosen (whether it is local or global).

Turning to the joint models, let us first observe that JOINTBESTFIRST outperforms BESTFIRST by a little over one and a half CoNLL point (from 72.96 up to 74.5). The performance gains come from large improvements in precision on MUC and B^3 (and a corresponding improvement in recall on $CEAF_e$). But the gains are much more significant with the structured version JOINTBESTFIRST^{struct} and the constrained version JOINTBESTFIRST^{constr}: there, the CoNLL score increases by more than 5 points, due to improvements of 3.5 in B^3, and of close to 11.8 in $CEAF_e$. On all three metrics, the gains are found in both recall and precision. Considering the nature of the $CEAF_e$ metric (calculated by finding the best matching between system and gold clusters), these results suggest that entities are better segmented in our joint models. Finally, our best model, JOINTBESTFIRST^{constr}, obtains a CoNLL score of 81.97, which is currently, up to our knowledge, the best achievement on gold mentions. By comparison, Chang *et al.* (2013) obtained a maximum score of 77.43 as previous highest score.

	MUC			B^3			$CEAF_e$			CoNLL
	P	R	F1	P	R	F1	P	R	F1	
Local Models										
BESTFIRST	89.75	77.03	82.90	84.23	65.95	73.98	52.03	76.68	61.99	72.96
PIPEBESTFIRST	83.63	84.21	83.92	66.98	74.93	70.73	66.5	64.97	65.73	73.46
JOINTBESTFIRST	91.95	76.89	83.75	88.16	65.70	75.29	53.07	82.12	64.47	74.50
Strutured Models										
BESTFIRST^{struct}	85.36	84.97	85.16	69.89	75.16	72.43	67.11	68.12	67.61	75.07
PIPEBESTFIRST^{struct}	84.61	82.19	84.90	67.94	75.56	71.55	97.54	65.99	66.76	74.40
JOINTBESTFIRST^{struct}	87.45	86.33	86.89	75.89	76.32	76.10	77.79	81.13	79.42	80.80
Constrained Models										
BESTFIRST^{constr}	86.44	85.42	85.93	73.87	76.33	75.08	67.28	69.93	68.58	76.53
PIPEBESTFIRST^{constr}	85.35	85.81	85.58	71.00	76.82	73.80	68.24	67.03	67.63	75.67
JOINTBESTFIRST^{constr}	88.40	87.04	**87.71**	78.49	78.14	**78.31**	77.93	81.92	**79.88**	**81.97**

Table 7.1: Coreference resolution on CoNLL-2012 Test Set English (gold mentions).

Anaphoricity detection We evaluate anaphoricity/discourse-new detection in the different models as follows: after resolving jointly or not anaphoricity and coreference, we label the first mention of each cluster as discourse new and all the rest as anaphoric and compare this with the gold partition. Results are reported in table 7.2. Notice that we do not report anaphoricity scores of pipeline models since they are exactly the same as the baseline local ananphoriciy model.

Before discussing the results, let us point out that gold mentions contain 22.7% of discourse-new mentions and 77.3% of discourse-old mentions. This distribution is biased towards anaphoric mentions, presumably more so than if singleton entities had been annotated in the CoNLL-2012 dataset.

Looking at table 7.2, we first observe that the two local models BESTFIRST and JOINTBESTFIRST have worse performance than the baseline anaphoricity model, and this even though JOINTBESTFIRST obtains better coreference results than the pipeline version. This suggests that anaphoricity should not be addressed locally without taking the whole context of the document. Structured and constrained pipeline models BESTFIRST^{struct} and BESTFIRST^{constr} do not worsen anaphoricity detection quality after coreference resolution, but do not improve it either (only minor increments are found).

On the contrary, joint models JOINTBESTFIRST^{struct} and JOINTBESTFIRST^{constr} show a very significant improvement over the local anaphoricity model, especially on discourse-new mentions. The accuracy achieved by the global joint model is very high compared to the other configuration (i.e., 98.52), and we saw that it also resulted in strong improvements on the coreference side. Overall, the large improvement in anaphoricity detection confirms that coreference entities are much better segmented in our joint model. We finally notice that, because of their deterministic aspect, the constraints of BESTFIRST^{constr} and JOINTBESTFIRST^{constr} slightly hinder the quality of anaphoricity detection compared to BESTFIRST^{struct} and JOINTBESTFIRST^{struct}.

	Discourse Old			Discourse New			overall
	P	R	F1	P	R	F1	accuracy
Baseline							
local anaphoricity model	93.18	94	93.59	79.37	77.04	78.19	90.09
Local Models							
BESTFIRST	96.42	82.78	89.08	60.97	89.75	72.61	84.39
JOINTBESTFIRST	99.8	83.48	90.92	64.34	99.45	78.13	87.16
Strutured Models							
BESTFIRST^{struct}	93.91	93.52	93.71	78.66	79.76	79.21	90.34
JOINTBESTFIRST^{struct}	99.66	98.41	**99.03**	94.91	98.87	**96.85**	**98.52**
Constrained Models							
BESTFIRST^{constr}	94.21	93.13	93.67	77.93	80.91	79.39	90.31
JOINTBESTFIRST^{constr}	99.45	97.95	98.7	93.5	98.19	95.79	98.01

Table 7.2: Anaphoricity detection on CoNLL-2012 Test Set English (gold mentions).

7.6 Related Work

A more complete review of the state of the art in coreference resolution was given in chapter 4. We detail here some aspects that are directly related to the work we present in this chapter.

7.6.1 Latent tree coreference models

The joint structured approach presented in this chapter directly extends recent work on latent tree structured models (Fernandes *et al.*, 2012; Chang *et al.*, 2013; Yu & Joachims, 2009; Björkelund & Kuhn, 2014). Given that it is difficult to globally optimize a clustering metric, these approaches instead optimize over the latent tree structures that represent the most informative links in the clusters. If these structures are very similar to those we use, they do not include anaphoricity information nor are used to jointly learn anaphoricity dectection and coreference resolution

This type of approaches breaks away from the standard mention-pair models (Soon *et al.*, 2001; Ng & Cardie, 2002b; Bengtson & Roth, 2008; Stoyanov *et al.*, 2010a; Björkelund & Farkas, 2012) and so-called ranking models (Denis & Baldridge, 2008; Rahman & Ng, 2011). Other structured output models to coreference include correlation clustering Finley & Joachims (2005) as well as various probabilistic graphical model-based approaches McCallum & Wellner (2004); Culotta *et al.* (2007). These learning models are more complex in that they also attempt to enforce transitivity. Other transitivity enforcing models use Integer Programming-based Klenner (2007); Denis & Baldridge (2009). Due to their much higher complexity, these global decoding schemes are used in combination with locally-trained models. Coreference resolution has also been framed as a (hyper)graph-cut problem Nicolae & Nicolae (2006); Cai & Strube (2010a). Several other models have attempted to break away from the mention pair representation altogether, trying to model cluster-mention or cluster-cluster relations Luo *et al.* (2004); Haghighi & Klein (2010); Rahman & Ng (2011); Stoyanov & Eisner (2012).

7.6.2 Anaphoricity detection

A number of previous work has attempted to model anaphoricity detection, and to combine it with coreference resolution.

Ng & Cardie (2002a) empirically show that the pipeline setting typically induces drops in coreference performance. Ng (2004b) shows that one can get improvement on coreference resolution, but this requires careful tuning of the anaphoricity classification threshold. Denis & Baldridge (2008) uses an anaphoricity classifier combined with a mention ranking model.

Previous joint approaches using ILP Denis & Baldridge (2007) or Markov Logic Net-

work Poon & Domingos (2008) (or more recently Bögel & Frank (2013)) have the drawback of formulating a problem which is NP-complete, and may be very time consuming. Rahman & Ng (2011) propose a local joint approach using ranking to decide whether a mention is discourse-new or linked to a previous entity.

Finally, we can find in Bengtson & Roth (2008) and Durrett *et al.* (2013) models that use anaphoricity as a feature in the coreference model.

7.7 Conclusion and Perspectives

We introducted a new model for jointly detecting anaphoricity and resolving coreference. Our model is based on latent tree structures that are straightforward to compute on a weighted graph. Our experiments on gold mentions show that both anaphoricity detection and coreference resolution are improved in the joint model compared to non-joint and pipeline models, leading to results that are significantly higher than state-of-the-art. The best of our models is a constrained version of the joint structured model and achieves a CoNLL score of 81.97 on gold mentions which is to date, up to our knowledge, the best score achieved on gold mentions.

The next step is to integrate joint model in a robust end-to-end coreference system running on raw text, in order to evaluate its performance on detected mentions and find out whether the improvements we obained here also apply in this more realistic setting. This work is carried out in chapter 8.

Chapter 8

From plain text to clusters: the complete task of coreference resolution

Considering the three previous chapters, we addressed our three objectives, that is to say: 1. to improve feature representation of documents in linear models (see chapter 5), 2. to study structured approaches to coreference resolution, processing information globally at the document level (see Chapter 6), and 3. to define a joint approach to coreference anaphoricity, which improves the accuracy on both tasks (see Chapter 7). For more stability and reproducibility of the results, we evaluated our coreference resolver on gold mention on the CoNLL-2012 corpus (gold mention are the annotated mentions and do not contain mentions without coreference link). For our work to be exhaustive, we also need to see how the different components behave when inserted into an end-to-end architecture: such a system takes raw text in input and outputs coreference partitions. The crucial question is now to know whether we still get the improvements we got before in this more realistic conditions, and how our complete system compares to the state of the art.

In previous chapters, we carried out our experiments on gold mentions in order to avoid introducing too many parameters in the model and a bias due to system mentions. In this chapter, we build a complete system that processes raw text and output coreference clusters. Compared to the preliminary models tested on gold mentions, an important additional part in the system is the mention dectection module, that is, the function that extracts mention spans from raw text. The other module we include in the architecture is a little bit unconventional for state-of-the-art systems: it is the singleton detector. The purpose of a singleton detector is to reduce the clustering possibilities by isolating some clusters that contain only one mention (or singletons), and letting the general coreference model processing the rest of the mentions. If we suppose that the singleton detector is perfect, this amounts to using the model trained to solve coreference on gold mentions (recall that gold mentions are mentions that have at least one coreferent counterpart). The more singleton detection is accurate, the more our previous experiments can be considered as "realistic".

8.1 Introduction

An end-to-end coreference resolver takes raw text in input and outputs clusters. Several intermediate processing steps can be inserted in between for characterizing the raw document and extracting features for the coreference model: POS tagging, tokenizing, constituent and dependency parsing, named entity recognition, morphological analysis or gender and number tagging are useful information for the model. All this information can be employed to detect coreference links, and it can be extracted with state-of-the-art tools (e.g. parsers). However all this implies that the performance of a coreference resolver strongly depends on the quality of this information. CoNLL-2011 and 2012 Shared Tasks (Pradhan *et al.*, 2011, 2012) introduced an annotated corpus containing all this pre-processed information so that all coreference resolvers can be compared on the same basis. But coreference annotations only include mentions with coreference links and exclude so-called singletons. Consequently, there is still a task we need to deal with: mention detection. Ideally, we would like to extract only the gold mentions, but knowing if a mention is a singleton is a global information on the coreference structure of the document. So we have to design a robust architecture that can be trained on a noisy set of mentions.

This chapter is organized as follows: in section 8.2, we describe our module for detecting mentions in raw text. In addition to this module, a singleton detector is studied in section 8.3. We show that state-of-the-art singleton detection can be greatly improved with little effort by modeling potential coreference links or potential anaphoricity of the mentions. These two modules are then inserted into the complete architecture, which is described in section 8.4. Learning issues are discussed in that section. Numerical results and comparison with state of the art are achieved in section 8.5.

8.2 Detecting mentions

In chapters 5, 6 and 7, we carried out experiments on gold mentions, assuming that they were given. An important step in building an end-to-end system is to select the set of mentions to be clustered during coreference resolution. In other words, it is the problem of detecting or extracting mentions from raw text. This step is crucial in a end-to-end coreference system, and it can have a significant impact on the performance of coreference resolution.

8.2.1 Detection methods and metrics

Mention detection involves several issues: first, the question of what kinds of mentions to take into account. All noun phrases are not necessarily mentions (for example, in “it is raining", *it* is not referential), and conversely, some mentions are not necessarily NPs (for example a verb in event coreference). Additionally, the annotation scheme can require to annotate elements of

a copula as coreferent (e.g., "**[he]** is **[the president of France]**") (muc, 1995) or to exclude them (Pradhan *et al.*, 2011, 2012). In practice, detecting mentions is not an obvious task and is often achieved through a set of heuristic rules extracting mentions from the parse tree (Lee *et al.*, 2011; Fernandes *et al.*, 2012; Lee *et al.*, 2013). Moreover, the choice of detecting method can have a significant impact on coreference resolvers: for instance, Kummerfeld *et al.* (2011) reports significant performance improvement with a same coreference resolver but additional filtering methods to "clean" detected mentions.

The second, more technical, problem we have when working on detected mentions is the choice of metrics. Indeed, it is necessary to match gold annotations and output partitions which are defined on a different set of mentions. Most metrics are not designed to work with detected mentions and several modifications have been proposed to deal with the full task (Cai & Strube, 2010b; Luo *et al.*, 2014; Pradhan *et al.*, 2014). In particular, Pradhan *et al.* (2014) carried out a re-evaluation of all systems participating in CoNLL-2012 Shared Task, involving some differences between new and original scores and ranking[1].

For these reasons, we preferred to carry out our previous experiments on gold mentions, so as to get rid of the uncertainty due to mention detection. In the present chapter, we extend the architecture of our system to work on raw text, and thus include mention detection in the evaluation.

8.2.2 A simple architecture for mention detection

We now describe the mention detection module we implemented and determine its quality compared to annotated data.

The architecture of our mention detection system combine several ideas we can find in the literature. The main idea is to start with the largest set of potential mentions and reduce it be combining several accurate deterministic and statistical filters. Indeed, we need to get rid of spurious mentions such as non-referential pronouns, or non annotated mentions (such as those appearing in copula for CoNLL-2012 Shared Task Corpus). Our detection procedure is the following:

1. Let $\mathcal{M}$ be the set of detected mentions. It is initially defined as the set of all NPs (noun phrases), PRP$s (possessive pronouns) extracted from the parse tree, and named entities which are included in larger NPs [2]. This is basically the detection procedure applied by Fernandes *et al.* (2012). This method has a large coverage on mentions, but introduces many spurious mentions. If not combined with further filters (as in Fernandes *et al.* (2012)), it can produce bad results.

[1] Hopefully, the top performers remained the same in the new ranking.

[2] Notice that here we do not extract verb mentions. Future work will include developing an accurate procedure for detecting this kind of mentions.

2. Only keep maximal NP projection in $\mathcal{M}$: remove NPs included in larger NPs whenever they have the same syntactic head (this step does not apply to named entities detected above). This kind of rules is applied in Durrett & Klein (2013); Lee *et al.* (2011, 2013). This first filter eliminates NPs that are very likely to be spurious mentions.

3. We next apply a set of heuristics, many of them inspired from Kummerfeld *et al.* (2011) and Lee *et al.* (2013). Specifically, we remove from $\mathcal{M}$:

   - Named entities tagged "PERCENT", "MONEY", "QUANTITY" or other than year "CARDINAL".
   - Second parts of appositions (e.g. The president of the US, Barack Obama)[3].
   - Attributes signaled by copular verbs.
   - Specific cases of *you* as in "you know" or "you can".
   - Quantified expressions: "all of us/you/them", "both of [...]", "some of [...]", "many of [...]".
   - Non words like "mm", "hmm", "ahem", "um".

4. Remove non referential pronouns it from $\mathcal{M}$. This step is more complicated, Lee *et al.* (2013) use a set of patterns on dependency parse tree to remove non-referential it. In our system we use the statistical model developed by Bergsma & Yarowsky (2011), which outperforms previous models.

We can roughly evaluate the quality of our filters: on the development set, where 17,804 gold mentions are annotated (recall that only mentions with at least one coreference link are annotated) we apply the detection procedure and get 39,853 mentions. We only count 361 gold mentions rejected by the filters, and 22,049 detected mentions that have no coreference link. So the ratio of gold mentions over detected mentions is about 43.77%, which means that we are a bit far from being in the same conditions as working on gold mentions.

In fact, once the set of detected mentions is determined, there are still mentions that are likely to be alone in their coreference clusters, which can be detected by examining their context. That means that we can reduce further the set on mentions on which the coreference resolver will operate. This latter procedure is explained in Section 8.3 below.

[3]Though this filter can seem unnecessary because of the previous rule, in practice it can eliminate NPs that were not because of head detection errors.

8.3 Eliminating singletons

We evaluated our models on gold mentions in Chapters 5, 6 and 7. The choice in CoNLL-2011 and 2012 was to not include mentions without coreference link in the gold. Trying to eliminate these mentions from the set of detected mentions is thus a natural procedure. This is not only justified by the evaluation protocol we chose, but empirically motivated by the shape of coreference partitions.

In a document, clusters coreferring mentions have various sizes. A typical distribution emerges from empirical observations: many mentions do not corefer with others and there are very few entities. In fact, the distribution of entity sizes roughly looks like a power law. Màrquez *et al.* (2013) observe that most clusters contain a single mention, and that generally entities are small. Singletons (i.e., clusters of size one) are 62.88% of mentions and 86.25% of entities on the data they study[4]. More importantly, as pointed out by Recasens *et al.* (2013), clusters of size one (or **singletons**) account for half of the data: they report that 56% of clusters are singletons in the CoNLL-2012 corpus development set[5].

A clustering approach to coreference with a Dirichlet process prior that gives control on the number of entities was proposed by Daumé III & Marcu (2005). We could extend this approach by using Pitman-Yor processes (Pitman & Yor, 1997), which would allow us to constraint the size of clusters and deal with this particular distribution of size. Another idea would be to employ extended spectral graph cut algorithms such as the one proposed by Zhou *et al.* (2014) to reproduce power laws. Nevertheless, in order to stay consistent with the structured approach to coreference resolution, we will not go in this direction in this thesis.

Consequently, here we only focus on separating singletons from the rest of clusters. Put differently, we want to detect mentions that have no coreference link with other mentions (we will call both singleton the entity and the single mention it contains). As mentioned by Recasens *et al.* (2013), this approach is justified empirically by the high proportion of singletons in a text document, whatever the category of document. Thus detecting singletons can be seen as a new task related to coreference resolution. It is not a preprocessing task because classifying a mention as singleton is a global choice on the document, nor a consequence of coreference resolution because some singletons are easy to detect. From the point of view of algorithmic complexity, preprocessing the text by removing singletons accelerates resolvers having a quadratic complexity (e.g. pair models), and even more complex combinatorial systems such as resolution using ILP.

[4]The corpora used in this study comprise the English, Catalan, and Spanish data sets from the SemEval-2010 Task 1 on Multilingual Coreference Resolution (Recasens *et al.*, 2010).

[5]To compute this, they consider the *mention boundaries* given in annotations as the set of mentions.

8.3.1 A new coreference-related task

Based on the fact that many entities appear only once in a discourse, Recasens *et al.* (2013) introduce this new coreference-related task of detecting singletons. Being able to predict whether a mention is a singleton or not would help solving the coreference task on detected mentions, but should be performed carefully because each singleton detected is a global choice on coreference partitioning. This task is thus formulated as a binary classification task, which is less complex and can be addressed by simple models.

In the particular case of CoNLL-2011 and 2012 corpora, gold mentions are only mentions with at least one coreference link. Thus gold mentions exclude both singletons (i.e., mentions without coreference link) and spurious mentions (e.g., NP that are not referential). In this case, the binary classification task is less "clean" in the sense that singletons are put together with spurious mentions against gold mentions[6].

A good handling of this task has been proposed by Recasens *et al.* (2013) to improve coreference resolution: in a pipeline architecture, the singleton detector exclude some spurious mentions and singletons so that the coreference resolver does less mistakes when clustering the rest of the mentions. In their experiment, the resolver is Stanford's multi-pass sieves resolver (Lee *et al.*, 2011), and improvement only appears when the binary classifier is set with a certain confident threshold to avoid removing too many non-singletons. The total improvement reported is about 0.7% in CoNLL score.

8.3.2 Improving the base model

The features of the binary classifier defined by Recasens *et al.* (2013) are inspired from discourse theory (e.g. Grosz *et al.* (1995); Walker *et al.* (1998)): grammatical and semantic role of the mention and morphosyntactic properties are taken into account.

In this subsection, we show that the model can be significantly improved with very little effort. In fact, we just have to notice that singleton detection intersects with other coreference task: first (and obviously), with the task of detecting coreference links. Second, with anaphoricity detection. Indeed, if a mention is anaphoric (or discourse-old), it cannot be a singleton.

That being said, adding some coreference link detection features, and anaphoricity features might help the model detects non-singleton mentions. Because the task is binary, improving the accuracy on one category would have a direct impact on the accuracy on the other one. We define a binary classifier with the same base features as Recasens *et al.* (2013), and employ the following additional features:

- **Base features:** *internal morphosyntactic* (grammatical type, animacy, person, number,

[6]Indeed, Recasens *et al.* (2013) consider as singletons all NPs which are not gold mentions.

Model	Singleton			Coreferent		
	P	R	F1	P	R	F1
Base model	79.57	81.72	80.63	76.23	71.39	73.73
Additional features *(single model)*	83.97	86.61	85.27	82.23	77.79	79.47
Additional features *(one model per category)*	83.83	87.82	**85.83**	82.64	77.23	**79.84**

Table 8.1: Singleton detection (logistic regression), on CoNLL-2012 English Dev.

indefinite, quantified, number of modifiers), *grammatical role features* (position in sentence, syntactic links, in coordination) and *semantic environment* (negation, modality, under attitude verb).

- **Anaphoricity features:** number of words, speech/reflexive pronoun, possessive description, bare noun, contains "of", contains relative/wh- pronoun, position in sentence/text.

- **Coreference features:** only simple relation with other mentions (embedding, string match, head match, apposition, acronym).

We reproduce the exact same experimental conditions as Recasens *et al.* (2013): the classifier is a logistic regression model, trained on the CoNLL-2012 Train set by using all NPs as mentions and gold mentions as non-singletons. The model is evaluated on the development set. We additionally define two modes for training: in the first version, the classifier is trained on all the corpus, in the second, one model is assigned to each category of document.

We only report here the score in "standard" threshold (50% in the logistic regression model). The scores are precision, recall and F1-score for the two categories (singleton vs coreferent). Overall results are reported in table 8.1. We can see that we almost reproduce the same results as Recasens *et al.* (2013) with the base model, and that the additive set of features provides a significant improvement of 5 points in F1-score for singletons and 6 points in F1-scores for coreferents, which is more than we could expected a priori: the intuitions that using features from similar task would help detecting singletons is confirmed here. Notice that the classifiers trained per category performs slightly better than the single classifier[7].

Now let us look and the detailed results per category for our best model (additional features and training per category) are reported in table 8.2. It is noticeable that we get various behaviors according to the category of document we work in. For example, the balance of score is inverted between MZ (where the model tends to be good on singletons) and PT (where it is better on coreferents). This suggests that the classification threshold should be adjusted, carefully, per category. Indeed, one potential issue is that the singleton model may hurt the coreference resolver by having too high a recall and too low precision. This would result in a very mis-aligned set of system mentions compared to gold mentions.

[7]Having good accuracy in singleton detection makes the coreference experiments on gold mentions more realistic.

Category	Singleton			Coreferent		
	P	R	F1	P	R	F1
BC	81.01	82.03	81.52	78.68	77.52	78.1
BN	81.79	85.87	83.78	80.99	75.89	78.36
MZ	89.11	93.9	91.44	83.6	73.04	77.96
NW	86.51	89.0	87.74	74.99	70.38	72.61
PT	76.95	77.14	77.05	91.44	91.36	91.40
TC	74.44	84.98	79.36	88.83	80.36	84.38
WB	81.37	91.49	86.13	85.22	70.09	76.91

Table 8.2: Singleton detection (logistic regression), results by category, CoNLL-2012 English Dev.

8.4 Building the end-to-end resolver

Having defined our mention detection module and the singleton classifier, we should be able to set up our full architecture now. In this section we detail the complete end-to-end coreference resolver, reading raw text and building mention partitions. We integrate all previously detailed modules along with a pre-processing NLP module to define the complete architecture. Since we rely to a machine learning approach, model traning issues are discussed and potential bottleneck identified.

8.4.1 Pre-processing raw text

The end-to-end system takes raw text in input (in our case, English). Documents are pre-processed i.e., parsed and tagged. The corpus delivered by CoNLL-2011 and 2012 Shared Tasks are already preprocessed in order to not have a bias due to the quality of parsing or tagging in coreference evaluation. Specifically, the corpus provides:

- Part of speech tags: these come along tokenization of the text.
- Syntactic parsing: this part is essential for extracting mentions, as we essentially work on NPs.
- Lemma for each token.
- Word sense: word sense disambiguation can help solving coreference in some cases (e.g., a bass as a guitar or a fish)
- Speaker information (when relevant): this information is particularly important in dialogs where the speaker uses the “I” pronoun.
- Named entity tags: can help extracting mentions, and adding constrains on coreference

links (e.g., personal pronouns can corefer with PERSON but not LOCATION).

- Predicate-arguments tags: role information can be employed in features to model a kind of focus (implying that some antecedents are less "accessible" than others for a given anaphora).

In addition, gender and number data extracted by Bergsma & Lin (2006) is authorized in closed track evaluation. Gender and number have critical importance in coreference resolution as they provide highly relevant information for constraining coreference links. Apart from all the features described above, document category is implicitly given (this is relevant to mention it as soon as we train models by category of documents).

Though all this information would be computed by a state-of-the-art NLP preprocessing module (except the speaker information, which might be harder to infer from text) in a complete system, in order to give results consistent with those of the campaign, our pre-processing module is implicit in our experiment, that is to say we use the information provided by the corpus. Indeed, it becomes more difficult to compare systems if their quality also depends on the quality of the parser or tagger that was employed.

8.4.2 The complete architecture

The next step, once the processing is done, is to detect mentions. As detailed in section 8.2 above, this procedure uses information built by the preprocessing module (in particular, NP and named entity spans are used to identify mentions). Once extracted, we reduce the set of mentions by eliminating singletons in order to process data set as close as possible to gold mentions. The last step of the architecture is of course the coreference resolver itself, using one of the models we designed in the previous chapters. The complete pipeline is depicted in figure 8.1.

Depending on if we use a joint model or not, anaphoricity and coreference resolution can be separated. If they are, the pipeline is extended by two separate modules are illustrated in figure 8.1. The architecture is now complete, but we still need to discuss some model learning issues.

8.4.3 Learning with detected mentions

Only having access to gold mentions in the annotated corpus is not only a matter of how to evaluate coreference resolver. It also implies that we need to adapt the learning procedure of our models. Indeed, the mention detector and the singleton model are not perfect and consequently the coreference model has to deal with non-singleton or spurious mentions. In our

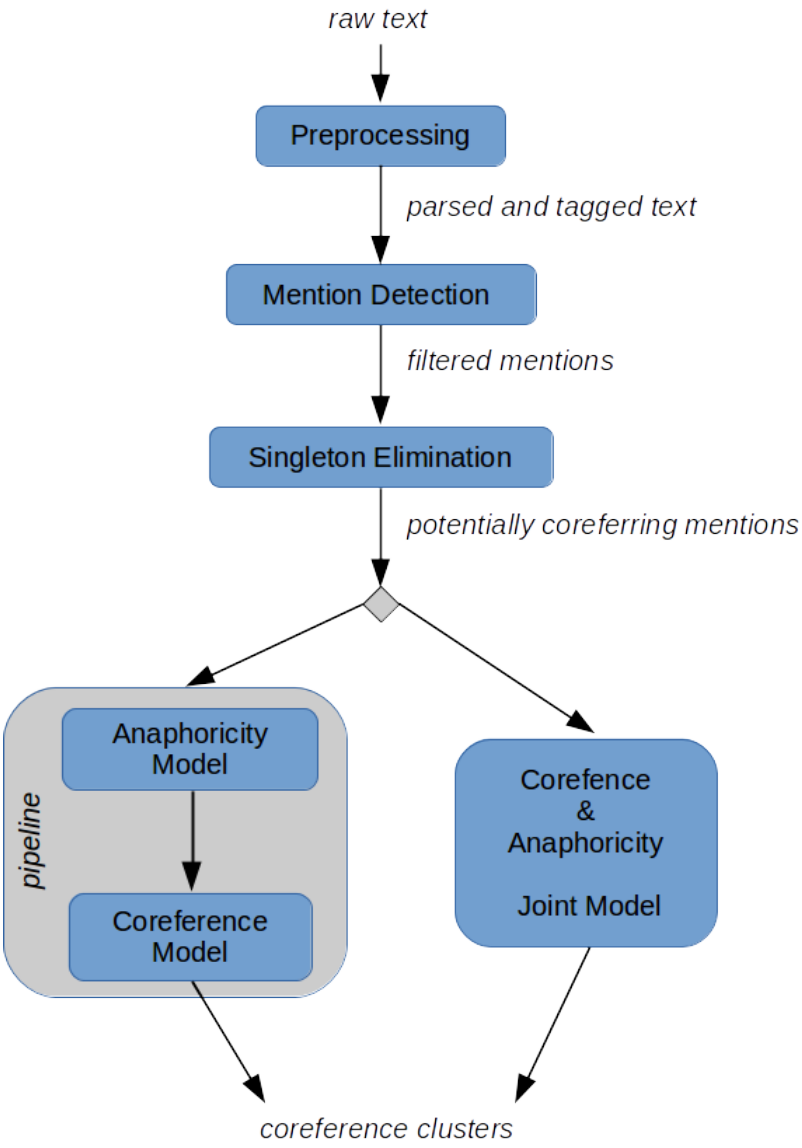

Figure 8.1: End-to-end coreference resolution architecture

case, employing directly models learned on gold mention led to very poor results. It is better for us to train the model on the mentions produced by our system.

Our resolver relies on three categories of model parameters for detecting singletons, anaphoricity and coreference. Learning these parameters is a sensitive issue: the performance and stability of the whole resolver will depend on the choices we will make here. We discuss several possibilities below.

Training the singleton model The singleton detector is the first statistical model employed in the pipeline. Our preliminary experiment showing improvements of the base model was carried out on all NPs. But in practice, because mention detection filters out many NPs, we prefer to train the singleton model on detected mentions. In that case, the training set is defined as the mentions detected on train documents, aligned with gold mentions. Put differently, our

gold singletons are the mentions that have been extracted by our mention detector but which are not annotated mentions. On the other hand gold coreference mentions those which have been extracted but which are not gold mentions.

As mentioned in section 8.3, if we use a 50% threshold, the accuracy of the singleton detector varies on the different document categories. A singleton which is not eliminated can still be alone in its coreference cluster after coreference resolution but the inverse is not possible: a mention classified as singleton is eliminated from coreference resolution. It may then better in this case to favor a balance towards precision for the singleton model, that is to say to be more confident when eliminating singletons.

Training the coreference model The first question is whether we should train our coreference models on gold mentions or detected mentions. Detected mentions are usually a better choice in an architecture without singleton elimination because the ratio of coreference links over all possible links is even more smaller, and directly using a model trained on gold mention may result in a to high recall of such links and very low precision.

Yet another problem arises: since we employ a singleton detector upstream, we should take into account the fact that its output is different from detected mentions. If the singleton model was perfect, we could directly use the parameters learned on gold mentions. However, we are not close enough to filter out all singletons, and the performance collapse completely if we use a model trained on gold mentions. We observed a certain stability of the singleton detector as it does not overfit training data, and gives accuracy results of about the same order as if we evaluate it on the training set or the development set. We thus use the output of the singleton model on the training set, and align it with gold annotations (i.e. we take the union of gold and detected mentions) to learn the coreference model.

Moreover, as discussed in Durrett & Klein (2013), adding undetected gold mentions in the coreference train set can improve the accuracy of a coreference resolver. We observed the same phenomenon with our coreference models, and consequently added undetected (or filtered out) gold mentions to the training set.

Training an anaphoricity detector If we do not use the joint model for anaphoricity and coreference, we need to specify the training set for an anaphoricity model. Theoretically, the anaphoricity model processes *all* mentions, and thus should be trained on detected mentions. However the singleton models also eliminates spurious mentions, and we found that the anaphoricity model performed better when trained after singleton elimination. Finally, to avoid introducing discrepancies between training sets, we employ the same mentions as for training the coreference model.

The training sets are now properly defined for our architecture. For training structured coreference models, we employ the perceptron learning algorithms described in section 6.3

(Chapter 6) and section 7.3 (Chapter 7). To optimize our singleton model thresholds, we try a range of values and select the best by trying to maximize the CoNLL score on the development set.

8.4.4 System description

The configurations we test in this chapter are defined as follows: the three first modules in the architecture (pre-processing, mention detection, singleton elimination) remain the same, and we plug in coreference models defined in the two previous chapters.

Recall that empirical evidence in 6 suggests that latent trees computed in a best-first fashion provide better models than, for instance, MST latent trees. In Chapter 7, we showed that structured models performed better that pipeline models. We restrain a bit the set of models we test here, but still make distinctions between local (i.e., pairwise), structured (using latent trees) and constrained (structured with must-link and cannot-link constraints) models.

Specifically, our baseline is the simple BESTFIRST pairwise model. JOINTBESTFIRST is the pairewise model addressing both coreference and anaphoricity for each mention. BESTFIRST^{struct} and JOINTBESTFIRST^{struct} are their structured versions, and BESTFIRST^{constr} and JOINTBESTFIRST^{constr} are their (structured) constrained versions.

Coreference Features We use the same common features as listed in Bengtson & Roth (2008); Rahman & Ng (2011): grammatical types and subtypes, string and substring match, apposition and copula, distance (number of separating mentions/sentences/words), gender and number match, synonymy/hypernymy and animacy, family name (based on lists), named entity types, syntactic features.

Anaphoricity Features Anaphoricity features are the same as in Chapter 7: number of words in the mention; binary features indicating if it is pronoun, speech pronoun, reflexive pronoun, proper name, definite description, quantified description, possessive description or bare noun; the position in text; if the mention is embedded in another mention; if the string/the head matches that of a preceding mention; if the mention is an apposition or acronym of a preceding mention (see Ng & Cardie (2002a); Ng (2004b); Denis & Baldridge (2008)).

Constraints We use the same constraints as in the previous chapter. Namely, must-links and cannot-links from sieve 1 of Lee *et al.* (2011), and additional cannot-links, using number, gender, and (un)animated mismatches, as well as i-within-i constraints. We add additional cannot-link constraints from the "statistical sieve" designed by Chen & Ng (2012), which consists in taking the full strings of mentions, and eliminating the pairs that are frequently not coreferent in the training data.

8.5 Experiments

In these experiments, our objective is to evaluate our coreference models when inserted in an end-to-end architecture. If mention detection and singleton classification was perfect, then our system would reproduce exactly the results we got on gold mentions. In other words, a potential bottleneck in our architecture can be one of these two modules.

Data As before, the models are evaluated on the CoNLL-2012 Shared Task English corpus, using "auto" preprocessing provided in the corpus. Additional information from wordNet and Bergsma & Lin (2006)'s gender and number database is used.

This time, we evaluate our models on the *detected mentions*. Considering the good scores we reported in Chapter 7, the focus is now to find out:

1. Whether the improvements we got on gold mentions with joint/structured/constrained transpose to detected mentions.

2. How our best model is ranked on detected mentions (in the new ranking provided by Pradhan *et al.* (2014)).

3. If there is a bottleneck in our architecture.

Evaluation We evaluate the models using the following three coreference resolution metrics: MUC Vilain *et al.* (1995), B^3 Bagga & Baldwin (1998a), and Entity-based CEAF (or $CEAF_e$) Luo (2005). We also report the CoNLL score (the unweighted average of the MUC, B^3 and $CEAF_e$ F1 scores). We employ the extended versions of the metrics defined in Pradhan *et al.* (2014) to deal with non-aligned partitions. Scores are aggregated by using micro-averaging, and before evaluating a partition against gold annotations, we remove singletons (either detected by the singleton model or isolated by the coreference model).

Settings All our coreference models are linear models, learned with the averaged perceptron algorithm (or its structured version) with 30 iterations on the training set which is enough to get stable scores. When learning structures, we use the max-loss learning mode, combined with the tree loss defined in Section 7.3 (Chapter 7).

The singleton model is the logistic regression model we evaluated before. We adjust its threshold for each category to optimize the CoNLL score obtained by the complete pipeline. This procedure is achieved on the development set.

8.5.1 Results and Discussion

Coreference resolution We report the final coreference results in Table 8.3. Compared to the figures in Chapter 7, the increments we get with more complex models are still here, but on a smaller scale. We obtain an improvement of 3 points in CoNLL score from the base model the joint and constrained structured model JOINTBESTFIRST^{constr} . If we compare to the score obtained by JOINTBESTFIRST^{struct}, we see that most of the improvement is due to the use of structure learning and joint resolution of coreference and anaphoricity.

The gain in joint resolution is more important on structured models (BESTFIRST^{struct} vs JOINTBESTFIRST^{struct}) than on pair models (BESTFIRST vs. JOINTBESTFIRST), which was also the case on gold mentions. Joint resolution of coreference and anaphoricity seems more beneficial when the whole (latent) structure of clusters is learned.

Our best model has a better balance in precision and recall than the base model, which tends to produce more fragmented partitions. Compared to the new ranking of the systems participating in CoNLL-2012 Shared Task (Pradhan *et al.*, 2014), our system would be ranked second with a score of 57.85. In that ranking, the first system (Fernandes *et al.*, 2012) obtains 60.7 and the second (Martschat *et al.*, 2012) 57.7. Other recents models (Björkelund & Kuhn, 2014; Durrett & Klein, 2013; Chang *et al.*, 2013) obtained scores of about the same order[8] as (Fernandes *et al.*, 2012).

As a result, even though our best model outperformed the existing on gold mentions, our end-to-end system is competitive with existing systems, but a bit below state of the art. A possible explanation for this is the difficult interaction between mention detection, singleton elimination and coreference resolution. Indeed, as we mentioned before, the first two modules in our system are the potential bottleneck. In practice our coreference models had a noticeable sensitivity to the threshold used in the singleton model. Additionally, as it is described above, the training scheme is also a bit complex and might not be optimal compared to the simple procedure we had on gold mentions.

When carrying out error analysis, we observed that many errors were due to the inclusion of non gold mentions in the clusters. The models tend to create clusters with mentions considered as singletons in the gold annotations. Because our detection module is one of the most restricted in terms of filters compared to existing systems, our investigation should be directed towards the interaction between the singleton model and the coreference models. When increasing the acceptance threshold too much, the singleton model starts to eliminate gold mentions and coreference results are rapidly hampered. On the other hand, if the threshold is too low, output partitions contain many "gold singletons" (i.e., mentions not in gold annotations) within larger clusters.

[8] One should be careful with the numbers given in the corresponding articles, because some were computed with the deprecated scoring method.

As a short conclusion of these experiments, the quality of the constrained and structured model (addressing anaphoricity and coreference jointly – JOINTBESTFIRST^{constr}) is now confirmed on gold mentions.

	MUC			B^3			$CEAF_e$			CoNLL
	P	R	F1	P	R	F1	P	R	F1	
Local Models										
BESTFIRST	78.54	57.71	66.53	69.55	39.95	50.75	54.17	41.73	47.14	54.81
JOINTBESTFIRST	79.04	58.18	67.02	70.04	39.18	50.25	52.42	45.02	48.44	55.90
Strutured Models										
BESTFIRST^{struct}	70.18	64.05	66.98	60.09	45.20	51.59	52.87	42.85	47.34	55.30
JOINTBESTFIRST^{struct}	69.15	67.04	68.08	55.19	51.86	53.47	54.20	45.8	49.65	57.07
Constrained Models										
BESTFIRST^{constr}	67.22	67.14	67.18	53.97	51.88	52.9	53.93	43.25	48.00	56.03
JOINTBESTFIRST^{constr}	69.4	68.22	**68.80**	55.63	53.54	**54.56**	54.83	46.29	**50.20**	**57.85**

Table 8.3: Coreference resolution on CoNLL-2012 Test Set English (detected mentions).

Mention detection One measure worth examining when evaluating a coreference resolver on system mention is the precision/recall of mention detection. However, because CoNLL corpus only contains gold mentions, singletons are not taken into account in the computation: precision and recall are calculated *after* coreference resolution and on gold mentions. To compute the score, system singletons are removed from the partition before comparing with annotations. We can be a bit skeptical about the usefulness of this measure for several reasons:

1. The score is sensitive to the coreference model we use.

2. It is possible not to count a spurious mention as an error in precision if it classified in a singleton (and thus eliminated) after resolution.

3. Singletons are not counted in the recall.

Despite these problems (all due to singletons not being annotated in gold), this biased precision/recall measure gives an idea of how the system behaves, and how good it is at extracting mentions. Our results are given in Table 8.4

	Mention Detection		
	P	R	F1
Local Models			
BESTFIRST	87.04	63.98	73.75
JOINTBESTFIRST	87.65	65.13	74.73
Structured Models			
BESTFIRST^{struct}	83.57	67.24	74.52
JOINTBESTFIRST^{struct}	84.64	68.46	75.7
Constrainted Models			
BESTFIRST^{constr}	76.49	73.59	75.01
JOINTBESTFIRST^{constr}	78.51	74.68	76.55

Table 8.4: Mention detection on CoNLL-2012 Test Set English.

Singleton elimination Finally, the last thing we need to look at is the scores obtained by our singleton classifier on detected mentions (recall that we evaluated the singleton model on all NPs). Because the threshold of the model has been adjusted for each category and with each combination with a coreference model, the results we give in Table 8.5 are only informative on the performance of singleton elimination.

Compared to the experiments on NPs, singleton elimination on detected mentions has a better precision on coreferring mentions. This is bit expected since NPs subsum detected mentions and the ratio of spurious mentions or singletons over coreferring mentions is greater. We cannot make any other comparison with state of the art as soon as the task of detecting singletons is new and a bit marginal in coreference resolution. In our experiments, the singleton model was an essential part which helped coreference models to be more accurate. In fact our coreference models lose about 4 CoNLL points if we do not use the singleton model.

Category	Singleton			Coreferent		
	P	R	F1	P	R	F1
BC	83.23	79.48	81.31	78.87	82.71	80.75
BN	84.10	84.54	84.32	80.83	80.31	80.57
MZ	89.72	92.72	91.19	82.18	75.97	78.95
NW	88.07	88.30	88.18	76.07	75.66	75.86
PT	77.62	77.00	77.31	91.16	91.43	91.30
TC	77.77	80.12	78.92	87.47	85.84	86.65
WB	82.60	89.35	85.84	83.51	74.13	78.54
overall	85.20	85.94	85.57	82.19	81.29	81.73

Table 8.5: Singleton elimination on CoNLL-2012 Test Set English.

8.6 Conlusion and outlooks

In this chapter, we designed an end-to-end coreference resolver, by adding two modules to the architecture we had in the previous chapters: mention detection and singleton elimination. Our mention detector was inspired from previous work and consists in a series of filters to remove NPs from detected mentions. Our singleton classifier is an improvement over the one introduced by Recasens *et al.* (2013): it is superior by more than 5 points of F1 score, just because we model potential coreference links and anaphoricity in the features. Singleton elimination is an essential part of our architecture.

When testing our coreference models defined in Chapter 7, we observed that structured learning provided improvements over pairwise models, and more importantly that our joint and constrained model also provided an improvement over existing latent trees models in the case of detected mentions. Our system achieves honorable results on CoNLL-2012: it would have been ranked second with 57.85 of CoNLL score. Nonetheless, improvement is still needed to reach the quality of the best current systems.

The weak side of our architecture lies in the learning procedure on detected mentions, and the interface between the singleton and the coreference models. One idea to overcome the difficulties that our architecture encounters when training the models would be to get more interaction between singleton and coreference in a single joint model. The model would be joint for singletons, anaphoricity and coreference. A direct way to implement that would be to use an ILP formulation of the problem or a backtracking algorithm, but we would lose the gain in complexity we have when using sparse tree structures. We thus need to develop both a well-suited structure and algorithm to address the three tasks jointly.

Besides, it is noticeable that we use quite simple learning models when relying on structured perceptrons for learning. Most of the gains were due to structural improvement, features and constraints. Now there is still room for model improvement on the learning side, in particular using non linear methods (e.g., structural SVM (Yu & Joachims, 2009)), large margin

method (e.g. structured PA/CW or AROW – see Chapter 3).

Chapter 9

Summary

In this thesis, we designed structured models for resolving coreference. Our main objective was to define a resolver that would process documents globally, achieving both coreference resolution and other related tasks, while having an acceptable sub-quadratic algorithimical complexity. To fulfill this objective, we defined three lines of research:

1. We improved the way features are used in linear models. In particular, we defined a method for improving pairwise linear models by building optimal hierarchies (see chapter 5). These hierarchies separate various kinds of pairs into disjoint feature spaces to avoid too different pairs to “overlap" during learning. Though a bit theoretical at first glance, this method is an extension of previous models separating different kinds of anaphora. We identified an elementary hierarchy (separation of mention pairs according to the grammatical type of each side of the pair) that provides a significant gain both on pairwise models and structured models. Indeed, our structured models employ this separation (see chapters 6, 7 and 8).

2. The second line of work consisted in finding a good structure for representing the coreference structure of a document and, more importantly, well suited to learning. We carried out detailed experiments in chapter 6, which identify a certain class of latent trees (computed by a best-first strategy) as a good structure for learning partitions globally. Surprisingly, maximum spanning tree, which have a better expressiveness in terms of possible structures do not perform better. Additionally, tree structures appear to be more suited to learning than denser graphs.

3. Finally, our last purpose was to modify the structured employed to represent coreference partitions in order to use additional information from coreference-related tasks. In particular, we designed a specific latent tree representation of both coreference clusters and anaphoricity of mentions. This allowed us to design an efficient model for solving these tasks. The model was further boosted by deterministic constraints allowing to capture more coreference links in a document. Up to our knowledge, our best model achieves the current best CoNLL score of 81.97 on gold mentions (see chapter 7) and is ranked just above the second system in the CoNLL-2012 Shared Task (see chapter 8).

When designing our end-to-end system, we addressed the problem of eliminating singletons (i.e., mentions without coreferring counterparts in the document). We improved state of the art significantly by modeling potential coreference links and anaphoricity in the features of our classifier. Singleton elimination guides the coreference model downstream but involves a rather complicated learning procedure for all the models. There is also a noticeable sensitivity of the whole architecture to the classification threshold of the singleton model.

In the future, as we did for anaphoricity, we plan to integrate singleton modeling into the latent structure representing coreference. It would simplify greatly both the architecture of the end-to-end system, and the learning procedure. Moreover, a joint resolution of coreference, anaphoricity and singletons might also benefit the three tasks. Apart from that, another way to improve our models would be to employ more advanced statistical learning methods such as large margin or kernel methods. Since the gains we had before were mostly due to structural redefinition or improvement, there are good odds to get more accurate models just by innovating on this technical side. Other possibilities are to employ different loss than the simple one we used for structured learning, for instance by penalized the links according to the type of the mentions. Learning can be achieved differently with partial early updates and with more refined structures to model different types of coreference interactions.

Bibliography

(1995). *MUC6 '95: Proceedings of the 6th conference on Message understanding*, Stroudsburg, PA, USA: Association for Computational Linguistics. 18, 33, 60, 123

Ariel, M. (1988). Referring and accessibility. *Journal of linguistics*, 24(1), pp. 65–87. 72

Asher, N. & Lascarides, A. (1998). Bridging. *Journal of Semantics*, 15(1), pp. 83–113. 22

Asher, N. & Lascarides, A. (2003). *Logics of conversation*. Cambridge University Press. 10

Bagga, A. & Baldwin, B. (1998a). Algorithms for scoring coreference chains. In: *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, Citeseer, pp. 563–6. 29, 83, 98, 115, 133

Bagga, A. & Baldwin, B. (1998b). Entity-based cross-document coreferencing using the vector space model. In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, Association for Computational Linguistics, pp. 79–85. 23

Bakir, G.H., Hofmann, T., Schölkopf, B., Smola, A.J., Taskar, B. & Vishwanathan, S.V.N. (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press, ISBN 0262026171. 43, 54

Bansal, M. & Klein, D. (2012). Coreference semantics from web features. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, Association for Computational Linguistics, pp. 389–398. 85

Bengtson, E. & Roth, D. (2008). Understanding the value of features for coreference resolution. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 294–303. 9, 47, 63, 67, 72, 80, 81, 88, 89, 93, 97, 106, 113, 114, 119, 120, 132

Bergsma, S. & Lin, D. (2006). Bootstrapping path-based pronoun resolution. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 33–40. 36, 68, 97, 115, 129, 133

Bergsma, S. & Yarowsky, D. (2011). Nada: A robust system for non-referential pronoun detection. In: *Anaphora Processing and Applications*, Springer, pp. 12–23. 68, 124

Bishop, C.M. (2007). Pattern recognition and machine learning (information science and statistics). 45

Björkelund, A. & Farkas, R. (2012). Data-driven multilingual coreference resolution using resolver stacking. In: *Joint Conference on EMNLP and CoNLL-Shared Task*, Association for Computational Linguistics, pp. 49–55. 63, 93, 102, 119

Björkelund, A. & Kuhn, J. (2014). Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. *ACL, Baltimore, MD, USA, June*. 66, 106, 108, 109, 110, 119, 134

Bobrow, D.G. (1964). A question-answering system for high school algebra word problems. In: *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pp. 591–614. 59

Bögel, T. & Frank, A. (2013). A joint inference architecture for global coreference clustering with anaphoricity. In: *Language Processing and Knowledge in the Web*, Springer, pp. 35–46. 120

Bos, J. (2008). Wide-coverage semantic analysis with boxer. In: *Proceedings of the 2008 Conference on Semantics in Text Processing*, Association for Computational Linguistics, pp. 277–286. 20

Boyd, S.P. & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. 50

Brennan, S.E., Friedman, M.W. & Pollard, C.J. (1987). A centering approach to pronouns. In: *Proceedings of the 25th annual meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp. 155–162. 10, 60

Broscheit, S., Poesio, M., Ponzetto, S.P., Rodriguez, K.J., Romano, L., Uryupina, O., Versley, Y. & Zanoli, R. (2010). Bart: A multilingual anaphora resolution system. In: *Proceedings of the 5th International Workshop on Semantic Evaluation*, Association for Computational Linguistics, pp. 104–107. 63

Cai, J. & Strube, M. (2010a). End-to-end coreference resolution via hypergraph partitioning. In: *Proceedings of the 23rd International Conference on Computational Linguistics*, Association for Computational Linguistics, pp. 143–151. 64, 72, 102, 119

Cai, J. & Strube, M. (2010b). Evaluation metrics for end-to-end coreference resolution systems. In: *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Association for Computational Linguistics, pp. 28–36. 31, 123

Carbonell, J.G. & Brown, R.D. (1988). Anaphora resolution: a multi-strategy approach. In: *Proceedings of the 12th conference on Computational linguistics-Volume 1*, Association for Computational Linguistics, pp. 96–101. 10, 20

Cardie, C., Wagstaff, K. *et al.* (1999). Noun phrase coreference as clustering. In: *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 82–89. 27

Chan, Y.S. & Roth, D. (2010). Exploiting background knowledge for relation extraction. In: *Proceedings of the 23rd International Conference on Computational Linguistics*, Association for Computational Linguistics, pp. 152–160. 12, 33

Chang, K.W., Samdani, R. & Roth, D. (2013). A constrained latent variable model for coreference resolution. In: *EMNLP*. 66, 88, 93, 94, 102, 106, 108, 110, 116, 119, 134

Chang, K.W., Samdani, R., Rozovskaya, A., Sammons, M. & Roth, D. (2012). Illinois-coref: The ui system in the conll-2012 shared task. In: *Joint Conference on EMNLP and CoNLL-Shared Task*, Association for Computational Linguistics, pp. 113–117. 88, 93

Chen, B., Su, J., Pan, S.J. & Tan, C.L. (2011). A unified event coreference resolution by integrating multiple resolvers. In: *IJCNLP*, pp. 102–110. 76

Chen, C. & Ng, V. (2012). Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In: *Joint Conference on EMNLP and CoNLL-Shared Task*, Association for Computational Linguistics, pp. 56–63. 11, 44, 62, 66, 67, 132

Chen, C. & Ng, V. (2013). Linguistically aware coreference evaluation metrics. In: *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pp. 1366–1374. 32, 39

Chinchor, N.A. (1998). Proceedings of the seventh message understanding conference (muc-7). p. 21 pages. 18, 60

Chu, Y.J. & Liu, T.H. (1965). On the shortest arborescence of a directed graph. *Science Sinica*, 14. 93, 109

Clark, H.H. (1975). Bridging. In: *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, Association for Computational Linguistics, pp. 169–174. 22

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, Association for Computational Linguistics, pp. 1–8. 46, 47, 54, 55, 88, 89, 96, 112

Collins, M. & Roark, B. (2004). Incremental parsing with the perceptron algorithm. In: *Pro-*

ceedings of the 42nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, p. 111. 47

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), pp. 273–297. 54

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. & Singer, Y. (2006). Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7, pp. 551–585. 49, 50, 55, 56, 73, 75, 96, 112

Crammer, K., Dredze, M. & Kulesza, A. (2009a). Multi-class confidence weighted algorithms. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, Association for Computational Linguistics, pp. 496–504. 53

Crammer, K., Dredze, M. & Pereira, F. (2008). Exact convex confidence-weighted learning. In: *Advances in Neural Information Processing Systems*, pp. 345–352. 52, 53

Crammer, K., Dredze, M. & Pereira, F. (2012). Confidence-weighted linear classification for text categorization. *The Journal of Machine Learning Research*, 98888, pp. 1891–1926. 52

Crammer, K., Kulesza, A. & Dredze, M. (2009b). Adaptive regularization of weight vectors. *Machine Learning*, pp. 1–33. 53

Culotta, A., Wick, M.L. & McCallum, A. (2007). First-order probabilistic models for coreference resolution. In: *HLT-NAACL*, pp. 81–88. 11, 61, 65, 102, 119

Dash, M. & Liu, H. (1997). Feature selection for classification. *Intelligent data analysis*, 1(3), pp. 131–156. 68

Daumé III, H. & Marcu, D. (2005). A bayesian model for supervised clustering with the dirichlet process prior. *The Journal of Machine Learning Research*, 6, pp. 1551–1577. 65, 125

Denis, P. & Baldridge, J. (2007). Joint determination of anaphoricity and coreference resolution using integer programming. In: *HLT-NAACL*, pp. 236–243. 11, 13, 26, 33, 61, 106, 108, 119

Denis, P. & Baldridge, J. (2008). Specialized models and ranking for coreference resolution. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 660–669. 13, 20, 64, 72, 106, 114, 119, 132

Denis, P. & Baldridge, J. (2009). Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42(1), pp. 87–96. 63, 72, 102, 119

Dredze, M., Crammer, K. & Pereira, F. (2008). Confidence-weighted linear classification. In:

Proceedings of the 25th international conference on Machine learning, ACM, pp. 264–271. 51, 52

Durrett, G., Hall, D.L.W. & Klein, D. (2013). Decentralized entity-level modeling for coreference resolution. In: *ACL (1)*, pp. 114–124. 65, 106, 113, 120

Durrett, G. & Klein, D. (2013). Easy victories and uphill battles in coreference resolution. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 68, 124, 131, 134

Edmonds, J. (1965). Optimum branchings. *Journal of Research of the National Bureau of Standards*. 93, 109

Ekbal, A., Saha, S., Uryupina, O. & Poesio, M. (2011). Multiobjective simulated annealing based approach for feature selection in anaphora resolution. In: *Anaphora Processing and Applications*, Springer, pp. 47–58. 68

Fernandes, E.R., dos Santos, C.N. & Milidiú, R.L. (2012). Latent structure perceptron with feature induction for unrestricted coreference resolution. In: *Joint Conference on EMNLP and CoNLL-Shared Task*, Association for Computational Linguistics, pp. 41–48. 36, 49, 62, 66, 68, 88, 89, 93, 94, 96, 102, 106, 108, 109, 110, 112, 119, 123, 134

Finley, T. & Joachims, T. (2005). Supervised clustering with support vector machines. In: *Proceedings of the 22nd international conference on Machine learning*, ACM, pp. 217–224. 102, 119

Freund, Y. & Schapire, R.E. (1999). Large margin classification using the perceptron algorithm. *Machine learning*, 37(3), pp. 277–296. 49, 96, 112

Garnham, A. (2001). *Mental models and the interpretation of anaphora*. Psychology Press. 22

Grosz, B.J. & Sidner, C.L. (1986). Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3), pp. 175–204. 60

Grosz, B.J., Weinstein, S. & Joshi, A.K. (1995). Centering: A framework for modeling the local coherence of discourse. *Computational linguistics*, 21(2), pp. 203–225. 10, 126

Haghighi, A. & Klein, D. (2007). Unsupervised coreference resolution in a nonparametric bayesian model. In: *Annual meeting-Association for Computational Linguistics*, volume 45, p. 848. 11, 65, 67

Haghighi, A. & Klein, D. (2010). Coreference resolution in a modular, entity-centered model. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 385–393. 65, 102, 119

Hastie, T., Tibshirani, R., Friedman, J. & Franklin, J. (2005). The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2), pp. 83–85. 85

Hirst, G. (1981). *Anaphora in Natural Language Understanding: A Survey*. Berlin, Germany: Springer-Verlag. 10, 18

Hobbs, J.R. (1976). Resolving pronoun references. *Research Report 76-1*. 60

Hobbs, J.R. (1978). Resolving pronoun references. *Lingua*, 44(4), pp. 311–338. 10, 60, 66

Hou, Y., Markert, K. & Strube, M. (2013a). Cascading collective classification for bridging anaphora recognition using a rich linguistic feature set. In: *EMNLP*, pp. 814–820. 23

Hou, Y., Markert, K. & Strube, M. (2013b). Global inference for bridging anaphora resolution. In: *HLT-NAACL*, pp. 907–917. 23

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L. & Weischedel, R. (2006). Ontonotes: the 90% solution. In: *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, Association for Computational Linguistics, pp. 57–60. 33

Huang, J., Taylor, S.M., Smith, J.L., Fotiadis, K.A. & Giles, C.L. (2009). Solving the who's mark johnson puzzle: information extraction based cross document coreference. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, Association for Computational Linguistics, pp. 7–12. 27

Kamp, H. (1981). A theory of truth and semantic representation. *Formal semantics-the essential readings*, pp. 189–222. 20

Kamp, H. & Reyle, U. (1993). From discourse to the lexicon: Introduction to model theoretic semantics of natural language, formal logic and discourse representation theory. 10

Karttunen, L. (1969). Discourse referents. In: *Proceedings of the 1969 conference on Computational linguistics*, Association for Computational Linguistics, pp. 1–38. 20, 22

Kehler, A., Appelt, D.E., Taylor, L. & Simma, A. (2004). The (non) utility of predicate-argument frequencies for pronoun interpretation. In: *HLT-NAACL*, volume 4, pp. 289–296. 72

Klenner, M. (2007). Enforcing coherence on coreference sets. In: *Proceedings of RANLP 2007*. 63, 72, 102, 119

Kohavi, R. & John, G.H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1), pp. 273–324. 68

Kruskal, J.B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1), pp. 48–50. 91

Kummerfeld, J.K., Bansal, M., Burkett, D. & Klein, D. (2011). Mention detection: heuristics for the ontonotes annotations. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, pp. 102–106. 40, 123, 124

Lappin, S. & Leass, H.J. (1994). An algorithm for pronominal anaphora resolution. *Computational linguistics*, 20(4), pp. 535–561. 66

Lascarides, A. & Asher, N. (2007). Segmented discourse representation theory: Dynamic semantics with discourse structure. In: *Computing meaning*, Springer, pp. 87–124. 10

Lassalle, E. & Denis, P. (2011). Leveraging different meronym discovery methods for bridging resolution in french. In: *Anaphora Processing and Applications*, Springer, pp. 35–46. 23

Lassalle, E. & Denis, P. (2013a). Apprentissage d'une hiérarchie de modèles à paires spécialisés pour la résolution de la coréférence. In: *TALN 2013-20ème conférence du Traitement Automatique du Langage Naturel 2013*. 71

Lassalle, E. & Denis, P. (2013b). Improving pairwise coreference models through feature space hierarchy learning. In: *ACL 2013*. 71, 102

Lassalle, E. & Denis, P. (2015). Joint anaphoricity detection and coreference resolution with constrained latent structures. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*. 105

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M. & Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4), pp. 885–916. 66, 123, 124

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M. & Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, pp. 28–34. 11, 36, 60, 61, 65, 66, 87, 97, 114, 123, 124, 126, 132

Li, B. (2012). Learning to model multilingual unrestricted coreference in ontonotes. In: *Joint Conference on EMNLP and CoNLL-Shared Task*, Association for Computational Linguistics, pp. 129–135. 37

Luo, X. (2005). On coreference resolution performance metrics. In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 25–32. 29, 83, 98, 115, 133

Luo, X., Ittycheriah, A., Jing, H., Kambhatla, N. & Roukos, S. (2004). A mention-synchronous coreference resolution algorithm based on the bell tree. In: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, p. 135. 64, 102, 119

Luo, X., Pradhan, S., Recasens, M. & Hovy, E. (2014). An extension of blanc to system mentions. *Proceedings of ACL, Baltimore, Maryland, June*. 31, 123

Màrquez, L., Recasens, M. & Sapena, E. (2013). Coreference resolution: an empirical study based on semeval-2010 shared task 1. *Language resources and evaluation*, 47(3), pp. 661–694. 31, 125

Martschat, S. (2013). Multigraph clustering for unsupervised coreference resolution. *ACL 2013*, p. 81. 64, 66

Martschat, S., Cai, J., Broscheit, S., Mújdricza-Maydt, E. & Strube, M. (2012). A multigraph model for coreference resolution. In: *Joint Conference on EMNLP and CoNLL-Shared Task*, Association for Computational Linguistics, pp. 100–106. 64, 66, 134

Matsushima, S., Shimizu, N., Yoshida, K., Ninomiya, T. & Nakagawa, H. (2010). Exact passive-aggressive algorithm for multiclass classification using support class. In: *SDM*, volume 10, pp. 303–314. 50

Matthews, A. & Chodorow, M.S. (1988). Pronoun resolution in two-clause sentences: Effects of ambiguity, antecedent location, and depth of embedding. *Journal of Memory and Language*, 27(3), pp. 245–260. 66

McCallum, A. & Wellner, B. (2004). Conditional models of identity uncertainty with application to noun coreference. In: *Proceedings of NIPS 2004*. 65, 102, 119

McCarthy, J.F. & Lehnert, W.G. (1995). Using decision trees for coreference resolution. In: *IJCAI*, pp. 1050–1055. 62, 72, 81, 88, 102

McDonald, R., Pereira, F., Ribarov, K. & Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 523–530. 55

Mejer, A. & Crammer, K. (2010). Confidence in structured-prediction using confidence-

weighted models. In: *Proceedings of the 2010 conference on empirical methods in natural language processing*, Association for Computational Linguistics, pp. 971–981. 56

Mejer, A. & Crammer, K. (2011). Confidence estimation in structured prediction. *arXiv preprint arXiv:1111.1386*. 56

Miller, G.A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11), pp. 39–41. 68

Mitkov, R. (2002). *Anaphora resolution*, volume 134. Longman London. 18, 21, 59

Mitkov, R., Evans, R., Orăsan, C., Pekar, V. *et al.* (2007). Anaphora resolution: To what extent does it help nlp applications? In: *Anaphora: Analysis, Algorithms and Applications*, Springer, pp. 179–190. 12

Morton, T.S. (2000). Coreference for nlp applications. In: *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp. 173–180. 64, 72, 83, 102

Müller, C. & Strube, M. (2006). Multi-level annotation of linguistic data with mmax2. *Corpus technology and language pedagogy: New resources, new tools, new methods*, 3, pp. 197–214. 32

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1), pp. 32–38. 30

Ng, A.Y. (2004a). Feature selection, l1 vs. l2 regularization, and rotational invariance. In: *Proceedings of the twenty-first international conference on Machine learning*, ACM, p. 78. 44

Ng, V. (2003). *Machine learning for coreference resolution: Recent successes and future challenges*. Technical report, Cornell University. 11

Ng, V. (2004b). Learning noun phrase anaphoricity to improve coreference resolution: Issues in representation and optimization. In: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, p. 151. 67, 106, 107, 114, 119, 132

Ng, V. (2005). Supervised ranking for pronoun resolution: Some recent improvements. In: *Proceedings of the National Conference on Artificial Intelligence*, volume 20, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 1081. 61, 64, 72

Ng, V. (2008). Unsupervised models for coreference resolution. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 640–649. 11, 67

Ng, V. (2010). Supervised noun phrase coreference research: The first fifteen years. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*, Association for Computational Linguistics, pp. 1396–1411. 105

Ng, V. & Cardie, C. (2002a). Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In: *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, Association for Computational Linguistics, pp. 1–7. 106, 114, 119, 132

Ng, V. & Cardie, C. (2002b). Improving machine learning approaches to coreference resolution. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp. 104–111. 9, 21, 33, 61, 63, 64, 67, 72, 81, 88, 89, 102, 113, 114, 119

Nicolae, C. & Nicolae, G. (2006). Bestcut: A graph algorithm for coreference resolution. In: *Proceedings of the 2006 conference on empirical methods in natural language processing*, Association for Computational Linguistics, pp. 275–283. 64, 72, 102, 119

Novikoff, A. (1962). On convergence proofs on perceptrons. In: *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pp. 615–622. 49

Partee, B.H. (1973). Opacity, coreference, and pronouns. In: *Semantics of natural language*, Springer, pp. 415–441. 18

Passonneau, R. (2004). Computing reliability for coreference annotation. Proceedings of the Language Resources and Evaluation Conference (LREC 2004). 32

Peng, H., Long, F. & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8), pp. 1226–1238. 44, 68

Ping, W., Liu, Q. & Ihler, A. (2014). Marginal structured svm with hidden variables, pp. 190–198. 56

Pitman, J. & Yor, M. (1997). The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2), pp. 855–900. 125

Poesio, M., Mehta, R., Maroudas, A. & Hitzeman, J. (2004). Learning to resolve bridging references. In: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, p. 143. 23

Poesio, M., Ponzetto, S. & Versley, Y. (2011). Computational models of anaphora resolution: A survey. *Linguistic Issues in Language Technology*. 18, 59

Ponzetto, S.P. & Strube, M. (2006). Exploiting semantic role labeling, wordnet and wikipedia

for coreference resolution. In: *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, Association for Computational Linguistics, pp. 192–199. 68, 72

Poon, H. & Domingos, P. (2008). Joint unsupervised coreference resolution with markov logic. In: *Proceedings of the conference on empirical methods in natural language processing*, Association for Computational Linguistics, pp. 650–659. 11, 42, 61, 65, 67, 106, 120

Pradhan, S., Luo, X., Recasens, M., Hovy, E., Ng, V. & Strube, M. (2014). Scoring coreference partitions of predicted mentions: A reference implementation. In: *Proceedings of the ACL 2014 Conference Short Papers, Baltimore, Md*, pp. 22–27. 31, 32, 40, 61, 123, 133, 134

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O. & Zhang, Y. (2012). Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In: *Proceedings of the Joint Conference on EMNLP and CoNLL: Shared Task*, pp. 1–40. 31, 33, 61, 68, 83, 97, 98, 115, 122, 123

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R. & Xue, N. (2011). Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, pp. 1–27. 31, 33, 61, 122, 123

Prim, R. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36, pp. 1389–1401. 91

Rahman, A. & Ng, V. (2011). Narrowing the modeling gap: A cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research*, 40(1), pp. 469–521. 20, 26, 65, 67, 80, 97, 102, 106, 108, 114, 119, 120, 132

Rand, W.M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336), pp. 846–850. 28

Ratinov, L. & Roth, D. (2012). Learning-based multi-sieve co-reference resolution with knowledge. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, pp. 1234–1244. 11

Recasens, M., de Marneffe, M.C. & Potts, C. (2013). The life and death of discourse entities: Identifying singleton mentions. In: *Proceedings of NAACL-HLT*, pp. 627–633. 26, 31, 36, 125, 126, 127, 138

Recasens, M. & Hovy, E. (2009). A deeper look into features for coreference resolution. In: *Anaphora Processing and Applications*, Springer, pp. 29–42. 9, 67

Recasens, M. & Hovy, E. (2011). Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17(04), pp. 485–510. 29, 30, 83

Recasens, M., Màrquez, L., Sapena, E., Martí, M.A., Taulé, M., Hoste, V., Poesio, M. & Versley, Y. (2010). Semeval-2010 task 1: Coreference resolution in multiple languages. In: *Proceedings of the 5th International Workshop on Semantic Evaluation*, Association for Computational Linguistics, pp. 1–8. 125

Rich, E. & LuperFoy, S. (1988). An architecture for anaphora resolution. In: *Proceedings of the second conference on Applied natural language processing*, Association for Computational Linguistics, pp. 18–24. 10

Richardson, M. & Domingos, P. (2006). Markov logic networks. *Machine learning*, 62(1-2), pp. 107–136. 61

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), p. 386. 47, 48

Song, Y., Jiang, J., Zhao, W.X., Li, S. & Wang, H. (2012). Joint learning for coreference resolution with markov logic. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, pp. 1245–1254. 65, 68

Soon, W.M., Ng, H.T. & Lim, D.C.Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4), pp. 521–544. 9, 11, 21, 33, 61, 62, 67, 72, 81, 87, 88, 89, 102, 119

Stoyanov, V., Babbar, U., Gupta, P. & Cardie, C. (2011). Reconciling ontonotes: Unrestricted coreference resolution in ontonotes with reconcile. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, pp. 122–126. 63

Stoyanov, V., Cardie, C., Gilbert, N., Riloff, E., Buttler, D. & Hysom, D. (2010a). Coreference resolution with reconcile. In: *Proceedings of the ACL 2010 Conference Short Papers*, Association for Computational Linguistics, pp. 156–161. 47, 63, 93, 102, 119

Stoyanov, V., Cardie, C., Gilbert, N., Riloff, E., Buttler, D. & Hysom, D. (2010b). Reconcile: A coreference resolution research platform. 63

Stoyanov, V. & Eisner, J. (2012). Easy-first coreference resolution. In: *COLING*, Citeseer, pp. 2519–2534. 65, 102, 119

Strube, M. (1998). Never look back: An alternative to centering. In: *Proceedings of the 17th*

international conference on Computational linguistics-Volume 2, Association for Computational Linguistics, pp. 1251–1257. 60

Tetreault, J.R. (1999). Analysis of syntax-based pronoun resolution methods. In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, Association for Computational Linguistics, pp. 602–605. 60

Tetreault, J.R. (2001). A corpus-based evaluation of centering and pronoun resolution. *Computational Linguistics*, 27(4), pp. 507–520. 66

Uryupina, O. (2004). Linguistically motivated sample selection for coreference resolution. In: *Proceedings of DAARC-2004*. 63, 72

Uryupina, O. & Poesio, M. (2012). Domain-specific vs. uniform modeling for coreference resolution. In: *LREC*, pp. 187–191. 33, 68

Uryupina, O., Poesio, M., Giuliano, C. & Tymoshenko, K. (2011a). Disambiguation and filtering methods in using web knowledge for coreference resolution. In: *FLAIRS Conference*. 68, 72

Uryupina, O., Saha, S., Ekbal, A. & Poesio, M. (2011b). Multi-metric optimization for coreference: The unitn/iitp/essex submission to the 2011 conll shared task. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, pp. 61–65. 68

Uzuner, O., Bodnari, A., Shen, S., Forbush, T., Pestian, J. & South, B.R. (2012). Evaluating the state of the art in coreference resolution for electronic medical records. *Journal of the American Medical Informatics Association*, pp. amiajnl–2011. 12

van Deemter, K. & Kibble, R. (2000). On coreferring: coreference in muc and related annotation schemes. *Computational Linguistics*, 26(4), pp. 629–637. 19

Vapnik, V.N. (1998). Statistical learning theory. 50

Versley, Y., Moschitti, A., Poesio, M. & Yang, X. (2008). Coreference systems based on kernels methods. In: *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, Association for Computational Linguistics, pp. 961–968. 63, 72

Vilain, M., Burger, J., Aberdeen, J., Connolly, D. & Hirschman, L. (1995). A model-theoretic coreference scoring scheme. In: *Proceedings of the 6th conference on Message understanding*, Association for Computational Linguistics, pp. 45–52. 28, 83, 98, 115, 133

Walker, C., Strassel, S., Medero, J. & Maeda, K. (2006). Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*. 33

Walker, M.A. (1989). Evaluating discourse processing algorithms. In: *ACL*, pp. 251–261. 60

Walker, M.A., Joshi, A.K. & Prince, E.F. (1998). Centering in naturally-occurring discourse: An overview. In: *In Centering in Discourse*, Citeseer. 10, 126

Wang, Z. & Vucetic, S. (2010). Online passive-aggressive algorithms on a budget. In: *International Conference on Artificial Intelligence and Statistics*, pp. 908–915. 54

Webber, B.L. (1978). *A formal approach to discourse anaphora.* Technical report, DTIC Document. 10, 18, 20

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. & Vapnik, V. (2000). Feature selection for svms. In: *NIPS*, volume 12, pp. 668–674. 68

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1), pp. 1–191. 59

Woods, W., Kaplan, R., Nash-Webber, B., Foundation, L.R. & Center, M.S. (1972). *The Lunar Sciences Natural Language Information System: Final Report*. vol. 1, Bolt Beranek and Newman. 59

Xiong, H., Song, L., Meng, F., Liu, Y., Liu, Q. & Lü, Y. (2011). Ets: an error tolerable system for coreference resolution. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, pp. 76–80. 37

Yang, Y., Xue, N. & Anick, P. (2011). A machine learning-based coreference detection system for ontonotes. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, pp. 117–121. 37

Yangy, X., Su, J., Zhou, G. & Tan, C.L. (2004). An np-cluster based approach to coreference resolution. In: *Proceedings of the 20th international conference on Computational Linguistics*, Association for Computational Linguistics, p. 226. 27

Yu, C.N.J. & Joachims, T. (2009). Learning structural svms with latent variables. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, pp. 1169–1176. 56, 65, 88, 89, 90, 93, 94, 102, 106, 108, 110, 119, 138

Zhekova, D. & Kübler, S. (2011). Ubiu: a robust system for resolving unrestricted coreference. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, pp. 112–116. 37

Zheng, J., Chapman, W.W., Crowley, R.S. & Savova, G.K. (2011). Coreference resolution: a

review of general methodologies and applications in the clinical domain. *Journal of biomedical informatics*, 44(6), pp. 1113–1122. 12

Zhou, X., Zhang, J. & Kulis, B. (2014). Power-law graph cuts. *arXiv preprint arXiv:1411.1971*. 125

Zhou, X., Han, H., Chankai, I., Prestrud, A. & Brooks, A. (2006). Approaches to text mining for clinical medical records. In: *Proceedings of the 2006 ACM symposium on Applied computing*, ACM, pp. 235–239. 12