Pontifícia Universidade Católica
do Rio de Janeiro

Adriel García Hernández

Coreference resolution for the English language

DISSERTAÇÃO DE MESTRADO

Dissertation presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Mestre em Informática

Advisor: Prof. Ruy Luiz Milidiú

Rio de Janeiro
April 2017

Pontifícia Universidade Católica
do Rio de Janeiro

Adriel García Hernández

Coreference resolution for the English language

Dissertation presented to the Programa de Pós-Graduação em Informática, of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Undersigned Examination Committee.

Prof. Ruy Luiz Milidiú
Advisor
Departamento de Informática – PUC-Rio

Prof. Marcus Vinicius Soledade Poggi de Aragão
Departamento de Informática – PUC-Rio

Prof. Leandro Guimarães Marques Alvim
UFRRJ

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico – PUC-Rio

Rio de Janeiro, April 26th, 2017

Adriel García Hernández

Graduated from University of Havana, Cuba in Computer Science. His research is focused in Machine Learning and Natural Language Processing.

Bibliographic data

García Hernández, Adriel

Coreference resolution for the English language / Adriel García Hernández; advisor: Ruy Luiz Milidiú. — Rio de Janeiro: PUC - Rio, Departamento de Informática, 2017.

62 f. : il. (color.); 29.7 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Departamento de Informática, 2017.

Inclui bibliografia.

1. Informática – Teses. 2. aprendizado de máquina. 3. processamento de linguagem natural. 4. resolução de correfenência. 5. modelo linear esparso. 6. indução de atributos.
I. Milidiú, Ruy Luiz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgements

To CNPq and CAPES and FAPERJ, for the financial support.

Thanks to PUC-Rio, for the support.

Thanks to my advisor PhD Ruy Luiz Milidiú for his support and contribution of knowledge.

Thanks to my family because without them it would not have been possible to reach this moment.

Thanks to my wife for supporting me and encouraging me to finish this project.

Thanks to all my friends who became part of my family.

Abstract

García Hernández, Adriel; Milidiú, Ruy Luiz (Advisor). **Coreference resolution for the English language**. Rio de Janeiro, 2017. 62p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

One of the problems found in natural language processing systems, is the difficulty to identify textual elements referring to the same entity, this task is called coreference. Solving this problem is an integral part of discourse comprehension since it allows language users to connect the pieces of speech information concerning to the same entity. Consequently, coreference resolution is a key task in natural language processing. Despite the large efforts of existing research, the current performance of coreference resolution systems has not reached a satisfactory level yet. In this work, we describe a structure learning system for unrestricted coreference resolution that explores two techniques: latent coreference trees and automatic entropy-guided feature induction. The latent tree modeling makes the learning problem computationally feasible, since it incorporates a relevant hidden structure. Additionally, using an automatic feature induction method, we can efciently build enhanced non-linear models using linear model learning algorithms, namely, the structured and sparse perceptron algorithm. We evaluate the system on the CoNLL-2012 Shared Task closed track data set, for the English portion. The proposed system obtains a 62.24% value on the competition's official score. This result is below the 65.73%, the state-of-the-art performance for this task. Nevertheless, our solution significantly reduces the time to obtain the clusters of a document, since, our system takes 0.35 seconds per document in the testing set, while in the state-of-the-art, it takes 5 seconds for each one.

Keywords

machine learning; natural language processing; coreference resolution; sparse linear model; feature induction; supervised machine learning.

Resumo

García Hernández, Adriel; Milidiú, Ruy Luiz. **Resolução de correferência para a língua inglesa**. Rio de Janeiro, 2017. 62p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Um dos problemas encontrados nos sistemas de processamento de linguagem natural é a dificuldade em identificar elementos textuais que se referem à mesma entidade. Este fenômeno é chamado de correferência. Resolver esse problema é parte integrante da compreensão do discurso, permitindo que os usuários da linguagem conectem as partes da informação de fala relativas à mesma entidade. Por conseguinte, a resolução de correferência é um importante foco de atenção no processamento da linguagem natural. Apesar da riqueza das pesquisas existentes, o desempenho atual dos sistemas de resolução de correferência ainda não atingiu um nível satisfatório. Neste trabalho, descrevemos um sistema de aprendizado estruturado para resolução de correferência sem restrições que explora duas técnicas: árvores de correferência latente e indução automática de atributos guiadas por entropia. A modelagem de árvore latente torna o problema de aprendizagem computacionalmente viável porque incorpora uma estrutura escondida relevante. Além disso, utilizando um método automático de indução de recursos, podemos construir eficientemente modelos não-lineares, usando algoritmos de aprendizado de modelo linear como, por exemplo, o algoritmo de perceptron estruturado e esparso. Nós avaliamos o sistema para textos em inglês, utilizando o conjunto de dados da CoNLL-2012 Shared Task. Para a língua inglesa, nosso sistema obteve um valor de 62.24% no score oficial dessa competição. Este resultado está abaixo do desempenho no estado da arte para esta tarefa que é de 65.73%. No entanto, nossa solução reduz significativamente o tempo de obtenção dos clusters dos documentos, pois, nosso sistema leva 0.35 segundos por documento no conjunto de testes, enquanto no estado da arte, leva 5 segundos para cada um.

Palavras–chave

aprendizado de máquina; processamento de linguagem natural; resolução de correfenência; modelo linear esparso; indução de atributos; aprendizado de máquina supervisionado

.

Table of contents

List of figures

List of tables

1
Introduction

When a person sees or hears a sentence, she makes full use of her experience and intelligence to understand it. This knowledge includes not only grammar, but also her knowledge of words, the context of the sentence, and most important, her understanding of the subject matter. To model this language understanding process in a computer, we need a program which combines grammar, semantics, and reasoning in an intimate way, concentrating on their interaction. The field of study that focuses on the interactions between human language and computers is called Natural Language Processing, or NLP for short.

A distinctive characteristic of this field is that, for each task, it usually exists a competition that establishes a well-defined problem setting, standard corpora, and evaluation metrics. The Conference on Natural Language Learning (CoNLL) Shared Tasks are examples of such competitions. Furthermore, they promote a significant number of advances in this research area.

Typically the NLP algorithms are based on machine learning (ML) algorithms. Instead of hand-coding large sets of rules, they can rely on ML to automatically learn these standards by analyzing a set of examples (i.e. a large corpus, like a book, down to a collection of sentences), and making a statical inference. In general, the more data analyzed, the more accurate the model will be.

Natural Language Processing includes structure learning (SL) problems, such as dependency parsing, part-of-speech (POS) tagging and quotation extraction. Dependency parsing is to identify a tree underlying a given sentence. In POS tagging, for a given input sentence, the prediction output is a sequence of tags. In quotation extraction, an input document is segmented into non-overlapping quotes that, additionally, are associated with their authors.

In this work, we are interested in another SL problem, the so called Coreference Resolution (CR), which consists in clustering mentions that are references to the same entity in a document. In this case, the output domain covers all possible clusters that can be formed by the entities, and hence, can be modeled as a structured prediction problem. For this effort, following a line of research in the *Laboratorio de Engenharia de Algoritmos e Redes Neurais* (LEARN) we base our investigation on the previous work Fernandes et al. (2014). Our emphasis in this document is on reporting our new modeling

strategies and the resulting new system for Coreference Resolution for the English language.

1.1 Coreference Resolution

Mentions are textual references to real-world entities or events. In a given document, mentions that refer to the same entity are called coreferring mentions and form a mention cluster. Coreference resolution is the task of identifying the mention clusters in a document and has been a core research topic in natural language processing. It has wide applications in question answering, machine translation, automatic summarization, and information extraction. Fernandes et al. (2014).

This problem has been carefully studied during the last decades. The first to evaluate it was Grishman and Sundheim (1996). Subsequently in the competition SemEval-2010 the goal was to assess and compare automatic coreference resolution systems for six different languages (Catalan, Dutch, English, German, Italian, and Spanish) in four evaluation settings and using four different metrics Recasens et al. (2010). Later the CoNLL-2011 Shared Task Pradhan et al. (2011) has been dedicated to the modeling of unrestricted coreference resolution for English text. The CoNLL-2012 Shared Task Pradhan et al. (2012) considering three languages: Arabic, Chinese, and English.

In the present work, we solve the problem proposed in Pradhan et al. (2012), but only for the English language. The unrestricted coreference resolution task consists of identifying the non-singleton mention clusters in a document. A singleton mention cluster is that which contains only one mention.

Our specific task here is to identify for each document its non-singleton mention clusters.

To solve this task, we divide it into three subtasks:

1. *Mention detection* - where the document mentions are predicted using a set of rules;

2. *Mention clustering* - where mentions clusters are predicted using a machine learning strategy;

3. *Singleton elimination* - where the clusters formed by a single mention are eliminated.

In Figure 1.1, we present an illustrative example annotated in the dataset used in Pradhan et al. (2012). First, eight mentions are detected and shown in bold. Next, we identified four mention clusters by tagging each mention with

different letters to indicate its cluster. Finally, clusters that contain only one mention are ignored, such as the one with "the Chinese people" as its unique mention.

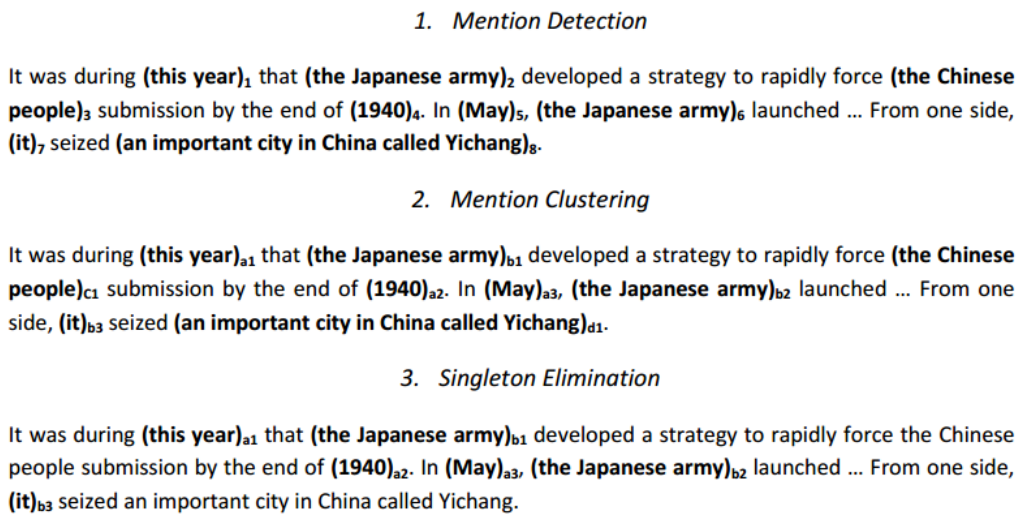

Figure 1.1: Unrestricted coreference resolution subtasks.

In this work, we follow the solution from Fernandes et al. (2014), which proposes an approach to unrestricted coreference resolution based on two essential modeling techniques: latent coreference trees and entropy-guided feature induction. Our approach relies on a graph whose nodes are the mentions in the given document. The arcs of this graph link mention pairs that are coreferent candidates.

The stages in which we can summarize the training, are the following:

1. *Mention detection* - where we build a graph node for each mention by adapting a predictor proposed by Santos and Carvalho (2011), and adding new rules;

2. *Candidate Pair Generation* - where we add a directed arc for each candidate coreferent mention pair by adapting the sieves proposed by Lee et al. (2013), and adding one filter;

3. *Basic Feature Setting* - where we set basic features that indicate whether an arc is likely to be connecting a coreferent pair by adapting the features used by Santos and Carvalho (2011);

4. *Feature Induction* - where we conjoin basic features to generate complex ones with high discriminating power by means of the entropy-guided feature induction method proposed by Fernandes (2012) and Milidiú et al. (2008);

5. *Coreference Tree Learning* - where we learn how to extract the trees that connect coreferent mentions in the graph, by applying a large margin, latent and sparse perceptron structure learning algorithm.

Predictor testing uses the same first three steps as in predictor training, followed by three further steps:

5. *Context Feature Setting* - where we set the values of the additional induced features selected at training;

6. *Coreference Tree Prediction* - where we apply a greedy algorithm to solve an optimal branching problem to find the maximum score coreference trees;

7. *Coreference Cluster Extraction* - where we extract the clusters of coreferring mentions from the coreference trees.

To evaluate our system we use the CoNLL-2012 Shared Task evaluation scheme, adopting the unweighted average of the MUC Vilain et al. (1995), B^3 Bagga and Baldwin (1998), and $CEAF_e$ Luo (2005) metrics.

1.2 Motivations and Goals

Coreference is a pervasive phenomenon in natural language. The problem lies at the intersection of syntax, semantics, and discourse. Coreference resolution is essential for natural language understanding and is necessary for many NLP applications, such as information extraction, question answering, and summarization. For this reason, we set ourselves the task of implementing a solution to this problem and achieving results close to state of the art.

To continue with the research line of our LEARN group, we selected the Fernandes et al. (2014) work, to carry out this implementation. Also, we noticed the possibility of making some variations to the perceptron algorithm to reduce the number of attributes to use during the prediction.

Main Goal

To implement a coreference resolution system, based on Fernandes et al. (2014), with generalization power similar to the state-of-the-art analyzers, that uses a reduced amount of attributes.

Specific Goals

To accomplish the main goal proposed above, we state the following subgoals:

1. to implement a predictor that extracts the mentions of the document, based on Santos and Carvalho (2011), and making some specifications for the English language;

2. to implement a predictor that determines the mention clusters, based on Fernandes et al. (2014);

3. to perform the induction procedure, using the Entropy-Guided Feature Generation algorithm Fernandes (2012); Milidiú et al. (2008); Santos and Milidiú (2009);

4. to perform the selection of the most informative features, using the Structured Sparse Perceptron Goldberg and Elhadad (2011);

5. to empirically evaluate and compare the proposed solution with state-of-the-art systems, using the English data set and the evaluation metric provided by the CoNLL Shared Task 2012 Pradhan et al. (2012).

1.3 Contributions

The main contributions of this dissertation are:

- the reduction of the number of attributes of the proposed model in Fernandes et al. (2014) in 92.54%;
- the decrease in 93% of the time to get the clusters of a document, compared to the time required by the state-of-the-art Clark and Manning (2016);
- the implementation of a predictor of mentions that improves the results achieved in Fernandes et al. (2014);
- the creation of a new technique of average for the structured perceptron;
- the simplification of the algorithm to solve an optimal branching problem to find the maximum score coreference trees;
- the implementation of the Entropy-Guided Structured Learning Framework proposed by Fernandes (2012) with Structured Sparse Perceptron and dropout technique;

– a system with competitive performance when compared against the state-of-the-art that attains 62.24% of accuracy when using the CoNLL Share Task 2012 metric Pradhan et al. (2012).

1.4 Dissertation Organization

The remainder of this document is organized as follows; In chapter 2, we review the background and related work on the coreference resolution task. In chapter 3, we describe our proposed learning system. Next, in chapter 4, we present the empirical evaluation of our approach. Finally, in chapter 5, we draw our conclusions and comment on interesting future work.

2
Related Works

Over the last two decades, many different machine learning–based approaches to coreference resolution have been proposed. Most of them use supervised learning and divide the task into two phases: the detection of potential mentions and the linking of mentions to form coreference clusters, that is, mention clustering Fernandes et al. (2014). Below we will present the techniques most used to address this problem.

2.1
Mention Detection

Most of the works that perform mention detection use a set of heuristics. The common approach consists of extracting all noun phrases (NP) from the parse tree and considering them to be candidate mentions Soon et al. (2001); Santos and Carvalho (2011); Haghighi and Klein (2010); Stoyanov et al. (2010); Chang et al. (2011); Lee et al. (2013); Bansal and Klein (2012); Sapena et al. (2013). A few works approach the mention detection task by training classifiers Bengtson and Roth (2008); Yuan et al. (2012).

2.2
Mention-Pair Classification

McCarthy and Lehnert (1995) were among the first to adopt a machine learning approach to resolving coreference. They evaluated a decision-tree-based system on the MUC-5 English Joint Venture corpus. The system was trained on all possible pairs in the training set, with eight features. The result outperformed an earlier heuristics-based system. Numerous systems were subsequently developed, and generally followed this paradigm.

One of the limitations acknowledged by the authors regarding their study is that the imbalance of the positive and negative training instances causes a bias towards classifying more negative pairs. Because all possible pairings of mentions are extracted, the negative instances far outnumbered the positive ones. An influential method to creating training instances to mitigate this problem was proposed in Soon et al. (2001). Positive instances were created from a mention and its immediate preceding mention that are coreferent. For every positive instance that involves mentions m_i and m_j, negative instances were created for each pair of mentions m_k and m_j, where $i < k < j$. A variant

of this method that differs slightly in the creation of positive instances was proposed in Ng and Cardie (2002), whereby the immediate preceding non-pronominal mentions is paired with a non-pronominal mention to create a positive instance. Another variant in the creation of negative instances was described in Ng and Cardie (2002). For every anaphoric mention m_j whose farthest antecedent mention to the left is m_i, a negative instance was created for each mention m_k such that $i < k < j$ and m_k and m_j are not coreferent.

Other methods in reducing the training instances focused on removing obvious negative instances to improve the training set balance or removing elusive positive instances to help the algorithm learn from "confident" pairs. Yang et al. (2003) removed mentions that violate gender, number or person agreement with the anaphor. Harabagiu et al. (2001) crafted rules manually to remove hard positive instances (such as those that require external knowledge) while preserving the coverage of clusters (based on the transitivity nature of the coreference relation) as much as possible. Ng and Cardie (2002) used a learner to exclude hard positive instances. Uryupina (2004) employed different methods in eliminating irrelevant or hard positive instances for pronoun, proper name, definite NP, and other types of anaphoric mentions. The number of features obtained from the training instances varies considerably, from a small set of eight McCarthy and Lehnert (1995) to nearly 40 Ng and Cardie (2002). Uryupina (2004) even reported 187 features. The features can either operate on one of the two mentions or both of them. Most of these features fall into one of the categories of lexical, syntactic, or semantic.

Lexical features mainly include string matching operations, such as exact match, substring match, and overlapping words. Syntactic features consist of grammatical roles, phrasal types, linguistic constraints like agreement and binding theory. Most of these syntactic features are derived from the parse trees in a heuristic manner. Semantic features usually involve consulting an external ontology, for example Miller (1995). Ng (2007) experimented with sophisticated semantic features but found limited performance gains, due to the difficulty in accurately computing these features. Bengtson and Roth (2008) evaluated the contributions of the features commonly used.

2.3 Entity-Mention

A common critique of the mention-pair model is that it cannot capture information beyond the mention pair. Consider a pair of a non-pronominal antecedent and a pronominal anaphor. The information that can be obtained from the two mentions to determine their coreferential status is very limited,

except for the gender and number agreements. Discarding these hard-to-resolve instances as discussed earlier may help the algorithms to learn from other strong evidence. However, this type of pair is a very frequent linguistic phenomenon.

In light of this shortcoming of the mention-pair model, Yangy et al. (2004) presented an approach to determining whether a noun phrase is coreferential with an existing (partial) coreferential cluster. They obtained better results on the GENIA data set Kim et al. (2003) than the mention-pair model using a decision tree system.

Training instances in an entity-mention model encompasses an anaphor and a cluster of preceding NPs. Instances are created similarly to the mention-pair model, i.e., for each positive instance, negative instances are created with the anaphora and its noncoreferential clusters.

In addition to features used in the mention-pair models describing the relationships between the anaphor and its antecedent, features encoding relationships between an anaphor and a partial cluster are added. These cluster-level features utilize first-order logic to expand upon the pairwise features. For example, the number agreement feature (whether the two mentions are both singular, or plural, or one is singular and the other plural) between the antecedent and anaphor in the mention-pair model can be transformed to the number agreement among the anaphor and all Yangy et al. (2004) or any Yang et al. (2008); Luo et al. (2004) of the NPs in the cluster.

2.4 Correlation Clustering

Finley and Joachims (2005); McCallum and Wellner (2004) formulate coreference resolution as a correlation clustering problem. However, they used different learning algorithms to predict the resulting clusters. The first formulated a supervised clustering method $SVM^{cluster}$ based on an SVM framework for learning structured outputs. The algorithm accepts a series of "training clusters," a series of sets of items and clusterings over that set. The method learns a similarity measure between item pairs to cluster future sets of items in the same fashion as the training clusters, and the second uses structured perceptron Collins (2002). Our system is also based on the structured perceptron; however, we use a large margin extension of this algorithm and use latent trees to represent each coreferring cluster. In Yu and Joachims (2009) they presented a framework and formulation for learning Structural SVMs with latent variables, applying this algorithm to the coreference resolution problem and comparing with the results obtained in

Collins (2002), achieving a significant improvement.

2.5 Coreference Tree

Yu and Joachims (2009); Fernandes et al. (2014) model the problem through coreference tree, although with different characteristics in trees. The former represents a document by a graph that relates the mentions by undirected arcs, whereby the clusters are shaped by undirected trees. Then they run any Maximum Spanning Tree algorithm such as Kruskal's algorithm Kruskal (1956) in the complete graph of mentions and output the clustering defined by the forest as the prediction. The latter represents a document using a directed graph since it relates the mentions by directed arcs, also creates a fictitious node that connects with all the mentions. Then they run a Chu-Liu-Edmonds Chu and Liu (1965); Edmonds (1967) algorithm to find the maximum score coreference tree, then each coreference tree son of the fictitious node, corresponds to a cluster of coreferring mentions. In both cases, the weight of each arc is the product of the vector of attributes of the arc and the vector of weights. They learn the vector of weights using structural SVMs and structured perceptron respectively.

2.6 Deep Learning

Recently Wiseman et al. (2015) introduced a simple, non-linear mention-ranking model for coreference resolution that attempts to learn distinct feature representations for anaphoricity detection and antecedent ranking. Clark and Manning (2016) improved the results of Wiseman et al. (2015) presented a coreference system that captures entity-level information with distributed representations of coreference cluster pairs. These learned, dense, high-dimensional feature vectors provide his cluster-ranking coreference model with a strong ability to distinguish beneficial cluster merges from harmful ones. The model is trained with a learning-to-search algorithm that allows it to learn how local decisions will affect the final coreference score.

2.7 Other systems

Other systems use global inference methods that combine classification and clustering in one step. Cai and Strube (2010); Cai et al. (2011); Sapena et al. (2013) present systems that implement global decision via hypergraph partitioning. Whereas Cai and Strube (2010); Cai et al. (2011) use a spectral

clustering algorithm to perform hypergraph partitioning, Sapena et al. (2013) use relaxation labeling for the resolution process. There are also approaches that perform global inference using integer linear programming to enforce consistency on the extracted coreference clusters Denis et al. (2007); Klenner (2007); Finkel and Manning (2008).

2.8 Chapter Conclusions

Several techniques have been applied to solve the problem of coreference resolution of a document. Here we reviewed the main topics referring to coreference resolution, showing the most used approaches to tackle this issue.

3
Coreference Resolution

Coreference resolution is the task of finding all expressions that refer to the same entity in a discourse Lee et al. (2013). To solve this problem, we follow the work of Fernandes et al. (2014), where the authors propose an approach to unrestricted coreference resolution that is based on two essential modeling techniques: latent coreference trees and entropy-guided feature induction. Its approach relies on a graph whose nodes are the mentions in the given document. The arcs of this graph link mention pairs that are coreferent candidates, and finally they use a structured and sparse perceptron to learn how to determine the correct cluster tree.

In Figure 3.1, we show the whole process of training. Firstly, we detect the mentions of each document, then create the candidate graphs, generating arcs among the most likely mentions of being in the same cluster. Later we generate many features for each arc. Finally, we use a structured and sparse perceptron to produce a model with an efficiency close to state of the art.

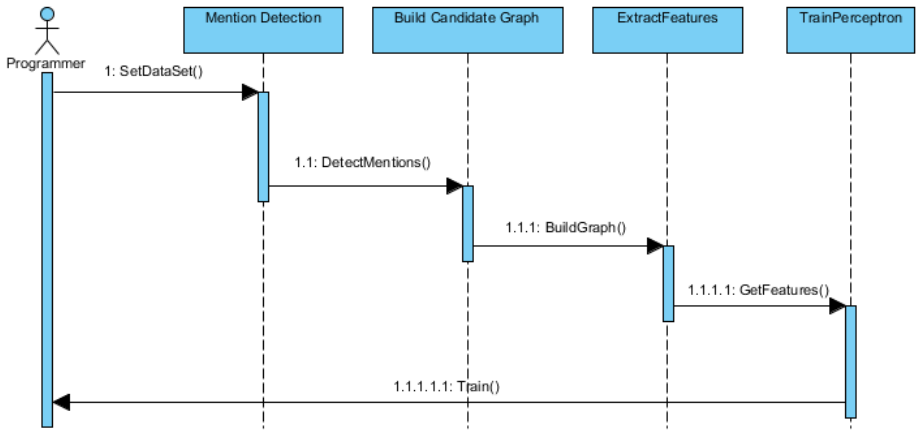

Figure 3.1: Train Sequence Diagram.

In this chapter, we will describe in detail each of these processes involved in solving our problems. First we give details of how the learning is performed with coreference pairs graphs. Then, we specify the edge-based scoring function that serves to discriminate candidates to coreference pairs graphs. Finally, we describe the structure perceptron and the corresponding extensions that need to be included in order to fulfill the proposed goals.

3.1 Mention Detection

Mentions detection is the first task to be solved in the problem of coreference resolution. As many authors have affirmed, the mention detection task is crucial to the performance of a coreference resolution system, because the mentions that are not selected in this first stage will not be considered during the rest of the process, so it is important at this moment to focus more on the recall than on precision.

In our system we implement an auxiliary predictor M given by

$$m = M(d)$$

where m is the list of detected mentions in the given document d. M is based on a particular set of rules that implement heuristics that have been proven effective in previous works Santos and Carvalho (2011), and we added some rules that showed improved recall and accuracy of this task. For a given document d, this predictor generates a list m of candidate mentions by including the following items:

1. noun phrases from the provided parse tree, not including nested NPs;
2. pronouns, even when they appear inside larger noun phrases;
3. named entities in the categories Person (PERSON), Organization (ORG), and Geo-Political Entity (GPE), even when they appear inside larger NPs;
4. possessive marks, even when they appear inside larger NPs;
5. demonstrative noun phrases, even when they appear inside larger NPs;
6. definite noun phrase, even when they appear inside larger NPs;
7. and noun phrases comprised only by proper nouns, even when they appear inside larger NPs.

The rules 2, 3, 4, 5, 6 and 7 have the effect of expanding the set of extracted noun phrases, thereby aligning the set of detected mentions more fully with the CoNLL-2012 annotated mentions. Our predictor M has the drawback of not including coreferring mentions of events that are in the CoNLL-2012 Shared Task data sets.

3.1.1 Mention Head

The concept of a mention is closely related to NPs in syntax. The reason for this relation is that CR at present focuses on entities and often ignores event coreference. As a consequence, finding the head of a mention generally corresponds to identifying the syntactic head of the corresponding noun phrase. In our system, we use a rule-based approach to solving this problem, similar to that proposed in Zhekova and Kübler (2013). We include a rule defining the head to be the last noun/pronoun in a sequence of nouns/pronouns. However, since we extract heads for mentions rather than for nested phrases, we consider context to account for the mixed directionality of English NPs, so we stop the search for the head when finding a preposition.

3.2 Candidate Pairs Graph Generation

Once completed the mentions detection subtask, it is possible to define the data set as follows:

$$D = \{(x, y)\}$$

where $x = (d, m)$, with d a unlabeled document, m its corresponding mention set, and y is the correct mentions clustering of d.

To represent a document, we create a graph $G(x)$ with its mentions, as it is proposed in Fernandes et al. (2014), this graph is called the *Candidate Pairs Graph*, and is constructed as follows: For each mention in m, a node is created in $G(x)$. Thus, an arc in $G(x)$ connects a pair of mentions in x. Ideally, we would consider the complete graph for each document. However, because the number of mentions may be large and because most mention pairs are not coreferent, we filter the arcs by using sieves from the method proposed by Lee et al. (2013). To further reduce the total number of arcs, we only consider forward arcs, which means that a directed arc (i, j) from mention i to mention j is only included in $G(x)$ if i appears before j in the document text. The arc that passes through any of the used sieves is called a **candidate arc**. Also, we refer to the two mentions of a candidate arc as a **candidate pair**. To complete $G(x)$, we add an artificial root node and connect it to each mention in x. These extra arcs are the **artificial arcs**. Therefore, given a mention pair (i, j), where i appears before j in the text, we create a directed arc $e = (i, j)$ if at least one of the following conditions holds:

1. Distance. The number of mentions between i and j is not greater than a given threshold.

2. Named Entity Alias. i and j has equals named entities, plus:
   - Person and GPE. The head word of one mention is part of the other mention.
   - Organization. The head word of one mention is contained in the other, or one is an acronym of the other.
3. Head Word Match. The head word of i matches the head word of j.
4. Shallow Discourse. This sieve comprises a set of rules proposed by Lee et al. (2013):
   - Singular, first person pronouns with the same speaker.
   - Singular, second person pronouns with the same speaker.
   - Singular, third person pronoun and a proper name with the same speaker.
5. Pronouns. j is a pronoun, and i has the same gender, number, and speaker as j, using the number and gender data from Bergsma and Lin (2006).

All these sieves are subject to the precondition that one mention is not embedded in another. We introduced this filter to achieve a better adaptation to the English portion of the CoNLL-2012 Shared Task data sets.

In Figure 3.2 we shown a possible graph for the Figure 1.1. Note that because we use a heuristic of sieves to generate the candidate arcs, will occur two types of problems. First, that a mention is not related to any other of the cluster to which it belongs, as is the case it_7 who is connected neither *the Japanese army*$_2$ or *the Japanese army*$_6$ which are the three mentions that conform the cluster. Second, there may be a candidate pair that does not belong to the same cluster, such as, *this year*$_1$ is connected with *the Japanese army*$_2$, but they are not in the same cluster.

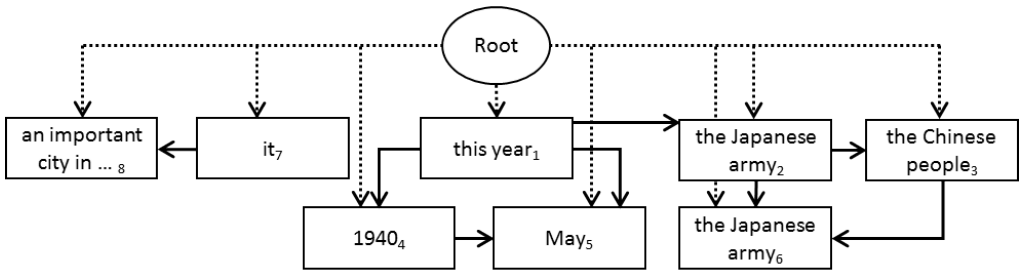

Figure 3.2: Candidate Pairs Graph.

In the first problem, the correct prediction of the cluster is impossible. However, such cases are rare, given that the recall of the used sieves is

approximately 90%. The second problem is the question to be solved by our learning algorithm. We use a structured and sparse perceptron that learns to eliminate from $G(x)$ the candidate arcs between mentions that are not in the same cluster. More precisely, the perceptron learns to extract a tree of $G(x)$, which is called **document tree** Fernandes et al. (2014). Removing the artificial root node and all its outgoing arcs, we splitting the document tree into a forest, each tree in this forest is called **coreference tree**, and its node set form a mentions cluster. Then we call the union of all mentions clusters extracted from each coreference tree in the forest as **document tree output**. As the proposed task in the CoNLL-2012 Shared Task does not consider the singleton clusters, we will only form the mentions clusters of coreference trees with more than one node.

Figure 3.3 shows a possible document tree extracted from Figure 3.2. Note that of all possible document tree, the document tree output of this is the most similar to the real clustering of the example. Precisely, this is the work of our perceptron, to construct a model, that for given a candidate pair graph $G(x)$ be able to extract from him, the better document tree.

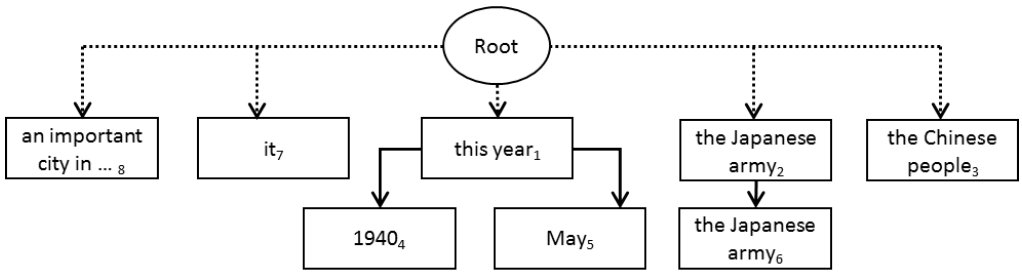

Figure 3.3: Document Tree.

3.3 Basic Feature Setting

To help our predictor identify the most tempting edges to be part of the document tree, we use basic features to describe a candidate arc. All of them give hints on the coreference strength of the mention pair connected by the arc. The features have been adapted from previously proposed features Fernandes et al. (2014); Ng and Cardie (2002); Sapena et al. (2010); Santos and Carvalho (2011)

Formally we define the feature vector that describes an arc $e = (i, j)$ who connect the mentions i and j as:

$$\Phi(x, e) = (\phi_1(x, e), ..., \phi_m(x, e))$$

where m is the number of features.

In Figure 3.4 we show how to obtain the features of each candidate pair (i, j). We consider features relative to mentions i and j, and features about the relation between mentions. Finally, we concatenate all the features to create the set of features.

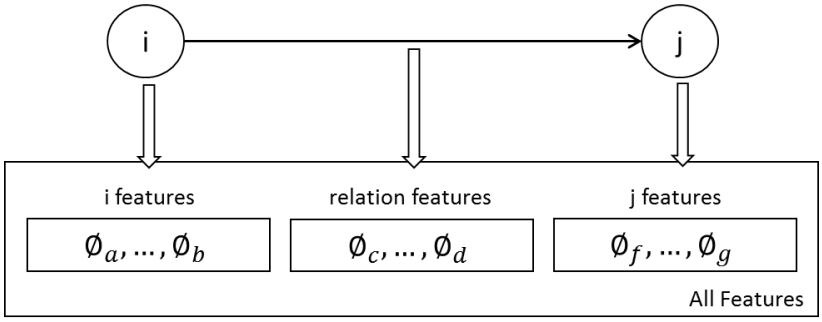

Figure 3.4: Candidate pair features representation.

These features provide lexical, syntactic, semantic, and positional information. In Table 3.1, Table 3.2, Table 3.3 and Table 3.4, we describe the 73 basic features used in our system. The *Id* column identifies each feature. The *Type* column indicates the value type of each feature. The # column indicates how many basic features correspond to each description.

Id	Description	Type	#
L1	Head word of i (j)	word	2
L2	String matching of i and j	bool	1
L3	String matching of the head words of i and j	bool	1
L4	Both i and j are pronouns and their strings match	ternary	1
L5	Both i and j are not pronouns and their string match	ternary	1
L6	Previous and next two words of i (j)	word	8
L7	Number of tokens in i (j)	integer	2
L8	Edit distance of head words of i and j	integer	1
L9	Edit distance of i and j after removing determiners	integer	1
L10	i (j) is a definite noun phrase	bool	2
L11	i (j) is a demonstrative noun phrase	bool	2
L12	The head word of both i and j are proper nouns	bool	1
L13	Both i and j are proper names and their strings match	ternary	1

Continuation of Table 3.1

Id	Description	Type	#
L14	Both i and j are proper names and their head word strings match	ternary	1
L15	i and j are relaxed match	bool	1
Total			26

Table 3.1: Lexical Basics Features

Id	Description	Type	#
Sy1	POS tag of the head word of i (j)	POS tag	2
Sy2	Previous and next two POS tags of i (j)	POS tag	8
Sy3	i (j) is a pronoun	bool	2
Sy4	Gender of i (j), if pronoun	f,m,n/u	2
Sy5	i and j are both pronouns and agree in gender	ternary	1
Sy6	i and j are both pronouns and agree in number	ternary	1
Sy7	i (j) is a proper name	bool	2
Sy8	i and j are both proper names	bool	1
Sy9	Previous and next predicate of i (j) within its sentence	verb	4
Sy10	i and j are pronouns and agree in number, gender, and person	ternary	1
Sy11	Noun phrase embedding level of i (j) in the syntactic parse	integer	2
Sy12	Number of embedded noun phrases in i (j)	integer	2
Sy13	Number of i (j), if pronoun	p,s/u	2
Total			30

Table 3.2: Syntactic Basics Features

Id	Description	Type	#
Se1	Baseline system prediction Santos and Carvalho (2011)	bool	1
Se2	Sense of the head word of i (j)	sense	2
Se3	Named entity type of i (j)	NE tag	2
Se4	i and j have the same named entity type	ternary	1

Continuation of Table 3.3

Id	Description	Type	#
Se5	Previous and next semantic roles for i (j) within its sentence	SRL tag	4
Se6	Concatenation of semantic roles of i and j for the same predicate, if they are in the same sentence	(SRL tag)^2	1
Se7	i and j have the same speaker	ternary	1
Se8	j is an alias of i Ng and Cardie (2002)	bool	1
Total			13

Table 3.3: Semantic Basics Features

Id	Description	Type	#
P1	Distance between i and j in number of sentences	integer	1
P2	Distance between i and j in number of mentions	integer	1
P3	Distance between i and j in number of person names, when i and j are both pronouns or one of them is a person name	integer	1
P4	One mention is in apposition to the other	bool	1
Total			4

Table 3.4: Positional Basics Features

3.4 Context Feature Induction

As a consequence of the fact that we use a learning algorithm based on a linear model, such as the structured and sparse perceptron, if we use only the basic features we do not capture enough information to represent coreference dependencies effectively. Conjoining basic features to derive new features is a common way to introduce nonlinear contextual patterns into linear models. In order to generate new nonlinear features, we use the entropy-guided feature induction (EFI) method Fernandes et al. (2014); Santos and Milidiú (2009); Santos and Carvalho (2011); Fernandes (2012). EFI automatically derives a set of basic feature conjunctions, which we denote **feature templates**. These templates are later used to generate the derived features, which comprise the

joint feature vectors $\Phi(x, e)$ described earlier.

3.4.1 Data set

First we need to modify our data set for a binary classifying task. Using all arcs in each of the candidate pair graphs, we obtain the arc data set $E = (\Psi(e), c(e))$, comprising the arc basic feature vectors $\Psi(e)$ along with their binary classification $c(e)$, where $c(e) = 1$ if the candidate pair in e are in the same cluster, and $c(e) = 0$ in the other case.

In Table 3.5, we depict an example of such a data set for the document in Figure 1.1 and the candidate pair graph in Figure 3.2. This example includes some features mentioned in the previous section Basic Feature Setting.

e		$\Psi(e)$					$c(e)$
i	j	L2	Se7	Sy12i	L3	P2	
1	2	0	1	0	0	0	0
1	4	0	1	0	0	2	1
1	5	0	0	0	0	3	1
2	3	0	0	1	0	0	0
2	6	1	1	1	1	2	1
3	6	0	0	1	0	2	0
4	5	0	0	0	0	0	1
7	8	0	0	0	0	0	0

Table 3.5: Data set for the example in Figure 1.1

The EFI method automatically generates feature templates for a structure learning problem by conjoining basic features that are highly discriminative. This method is based on the conditional entropy of arc label $c(e)$ given the basic features $\Psi(e)$.

3.4.2 Feature Template

For generating the templates, first we train a decision tree(DT) on the data set E. One of the most used algorithms for DT induction is C4.5 Quinlan (1992). We use Quinlan's C4.5 system to obtain the required entropy guided selected features.In Figure 3.2, we present a decision tree learned from a basic data set. Each internal node in the DT corresponds to a feature, each leaf node has a label value (0 or 1, in the binary case), and each edge is labeled with a value of the source node feature. Additionally, we eliminate template duplicates.

To extract feature templates, we use a depth-first traversal of the learned DT and is recursively defined as follows. For each internal node that is visited,

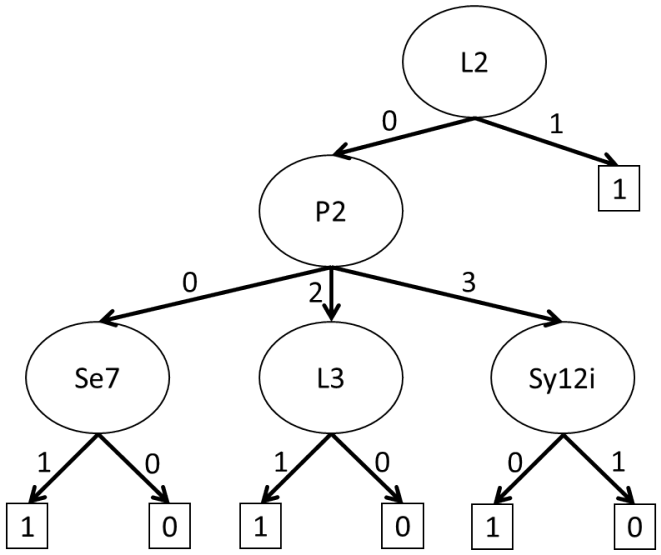

Figure 3.5: Decision Tree for Table 3.5.

a new template is created by conjoining the node feature with its parent template. Since we aim to generate feature templates as conjunctions of basic features, we not consider the feature values and the decision variable values in the DT. Thus, we do not make use of edge labels nor leaf nodes.

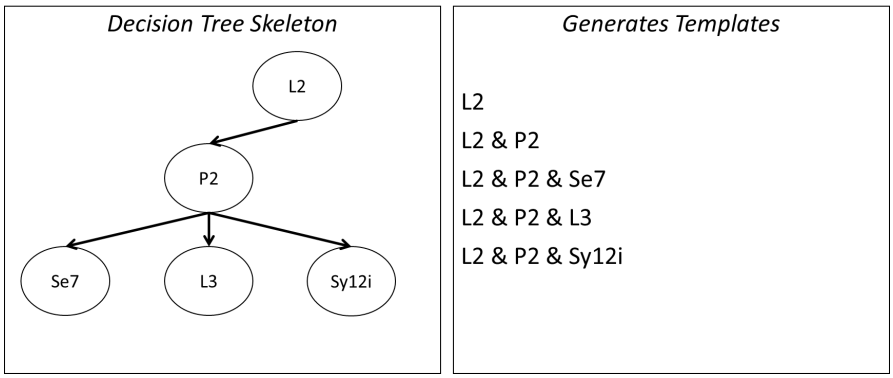

Figure 3.6: Features templates.

Figure 3.6 illustrates our method to generate the templates. The tree on the left side of this picture is the skeleton obtained from the decision tree in Figure 3.5, in it remains only nodes which are basic features with high discriminative power. On the right side of the figure are the generated templates.

This method is able to provide a very large number of templates. Hence, to limit the maximum template length, we use C4.5 pruned trees and additionally limit the maximum template length when traversing the DT.

3.4.3 Generated Features

For generate the binaries features, we employ the templates to induce all binary contextual features that occur in the data set E for the rows where $c(e) = 1$. For each template; we create several binary features, each one corresponding to the assignment of valid values to the template features. For instance, for the **L2 & P2 & Sy12i** template in Figure 3.6, and using the data set shown in Table 3.5, we will generate only three binary attributes, because there are three rows with $c(e) = 1$, one of this feature is:

$$\phi_m(x, e) = \begin{cases} 1 & if\ L2 = 0\ and\ P2 = 2\ and\ Sy12i = 0, \\ 0 & otherwise. \end{cases}$$

The derived feature vector $\Phi(x, e)$ is a binary vector whose 1-valued positions indicate the derived features that are active in arc e. These active derived features depend on the values of the basic features $Psi(e)$. The number of derived features is enormous because several combinations of basic feature values arise for each template. However, the number of active derived features in a specific arc is much smaller. In fact, for each arc, there is one active derived feature for each template; all others are set to zero.

3.5 Coreference Tree Learning

At this point, we have already presented all the necessary elements to introduce our learning algorithm. We must find a structure predictor F that assigns to its document input $x = (d, m)$ a high-quality document tree $\widehat{y}$ given by:

$$\widehat{y} = F(x)$$

We learn F from a training set D of correct input–output pairs. More specifically, the structured perceptron algorithm learns the weight vector w of a parameterized predictor given by:

$$\widehat{y} = F(x; w) = arg\, max_{h \in H(x)}\, s(x, h; w)$$

where $H(x)$ is the set of feasible outputs for x, that is, all document trees over the detected mentions m in $G(x)$, and s is a w-parameterized scoring function over the trees. The prediction $\widehat{y}$ is the solution of an optimization problem and is called the prediction problem. The objective function of this problem is given by s, it scores the candidate document trees for the given document.In

McDonald et al. (2005); Fernandes (2012); Fernandes et al. (2014), the $argmax$ combinatorial problem reduces to a maximum branching problem, which they solved using efficiently the Chu-Liu-Edmonds algorithm Chu and Liu (1965).

We construct the candidate pair graph $G(x)$ in a way that all edges go forward, because if exist an edge from i to j, then i has to appear in the document before j. This property ensures that in $G(x)$ there are no cycles. Then to solve the maximum branching problem, it is enough to use a greedy algorithm that selects for each node the most weighted edge that enters the node. In this way, we greatly simplify our problem. In the figure, we formally write this algorithm.

Input: $G(x)$ candidate pair graph
Output: y maximum branching tree
$y \leftarrow Nodes(G(x))$
for each node **in** $G(x)$
    $e \leftarrow MostWeightedEdge(G(x), node)$
    $AddEdge(y, e)$
return y

Figure 3.7: Greedy algorithm for maximum branching problem.

Only the weights of each arc are to be defined. Recall that a set of features $\Phi(x, e) = (\phi_1(x, e), ..., \phi_m(x, e))$ represents each arc, then using the vector w we define the the weight of each edges e as:

$$w(e) = \sum_{i=1}^{m} \phi_i(x, e) * w[i]$$

Summarizing, we train a structured and sparse perceptron to generate a vector of weights w, which helps to extract a document tree that has the best document tree output from a candidate pair graph $G(x)$, using the greedy algorithm in Figure 3.7 for solve the maximum branching problem.

During the rest of the chapter, we will formally explain our learning algorithm.

3.5.1 Structured Perceptron

In Collins (2002) was presented the structured perceptron. This algorithm combines the perceptron algorithm for learning linear classifiers with an inference algorithm and can be described as follows. First define a *joint feature function* $\Phi(x, y)$ that maps a training sample x and a candidate prediction y to a vector of length n. Then, its work as follows:

Input: $D = \{(x, y)\}$ binary data set
Output: w
$w \leftarrow 0$
while no convergence
  for each $(x, y) \in D$
    $\widehat{y} \leftarrow arg\, max_{h \in H(x)}\, s(x, h; w)$
    $w \leftarrow w + \Phi(x, y) - \Phi(x, \widehat{y})$
return w

Figure 3.8: Structure Perceptron algorithm.

If we look at Figure 3.8, we can see that we still have to define two important elements in this algorithm. Firstly we have no defined Φ for a document tree, but for an edge. Then the next predictor ingredient to be defined is the feature vector $\Phi(x, h)$. This vector is linearly decomposed as:

$$\Phi(x, h) = \sum_{e \in h} \Phi(x, e)$$

The second point to note is that in our dataset $D = \{(x, y)\}$ the outputs y are mention cluster, whereas our prediction $\widehat{y}$ is a document tree. Then there is no way to update the weight vector w. Thus, we assume that these tree is *latent*, then, we need use the latent structured perceptron algorithm Yu and Joachims (2009); Sun et al. (2009); Fernandes and Brefeld (2011); Fernandes et al. (2014) to train our models.

3.5.2 Latent Tree Learning

To modify our algorithm, we predict the constrained document tree $\widetilde{y}$ for the training instance (x, y), using a specialization of the document tree predictor, the constrained tree predictor $F_y(x, w)$. This predictor finds a maximum scoring document tree for x that follows the correct clustering y. Therefore, its search is constrained to a subset $H_y(x)$ contained in $H(x)$. The constrained tree predictor is given by:

$$\widetilde{y} = F_y(x; w) = arg\, max_{h \in H_y(x)}\, s(x, h; w)$$

The trees in $H_y(x)$ are subgraphs of the constrained candidate pairs graph $G_y(x)$, which is obtained by removing intercluster arcs regarding the correct clustering y from the candidate pairs graph $G(x)$. Regarding the artificial arcs, for each cluster in y, $G_y(x)$ includes only one arc that connects the artificial node to the first mention of the cluster. Hence, the constrained document tree

can only include arcs between mentions that lie in the same cluster plus one arc from the artificial root node to each cluster. We use the constrained document tree $\widetilde{y}$ as the ground truth in the current perceptron iteration. In Figure 3.9 we can appreciate the changes made to the structured perceptron of the Figure 3.8 to turn it into a latent structured perceptron.

Input: $D = \{(x, y)\}$ binary data set
Output: w
$w \leftarrow 0$
while no convergence
  for each $(x, y) \in D$
    $\widetilde{y} \leftarrow arg\, max_{h \in H_y(x)}\, s(x, h; w)$
    $\widehat{y} \leftarrow arg\, max_{h \in H(x)}\, s(x, h; w)$
    $w \leftarrow w + \Phi(x, \widetilde{y}) - \Phi(x, \widehat{y})$
return w

Figure 3.9: Latent Structure Perceptron algorithm.

3.5.3 Large margin and root loss

In this work, we use a large margin generalization of the structure perceptron Tsochantaridis et al. (2005); Fernandes (2012); Fernandes et al. (2014) that is based on the well known margin rescaling technique for structural support vector machines. We embeds a loss function into the prediction problem. The large margin predictor for a training example (x, y) is given by:

$$\widehat{y} = arg\, max_{h \in H(x)}\, s(x, h; w) + C * l_r(h, \widetilde{y})$$

where l_r is a non-negative loss function that measures how much a document tree h differs from the constrained document tree $\widetilde{y}$, which is the ground truth for the current iteration. The loss function measures the impurity in the predicted document tree, and C is a loss parameter tuned on the development set. In our model, we use a loss function proposed in Fernandes et al. (2014) that counts how many arcs are different in a given tree h compared to the current iteration ground truth $\widetilde{y}$, that is,

$$l_r(h, \widetilde{y}) = \sum_{(i,j) \notin h} 1[(i, j) \in \widetilde{y}] * (1 + r * 1[i = 0])$$

The loss function can be factored along arcs as follow, For each arc $(i, j) \in G(x)$, if $(i, j) \notin \widetilde{y}$, add C to the cost of the arc, and if in addition i is the fictitious node, the cost is increased by $C * r$.

This large margin is only applied during training because its purpose is to learn a model that separates the ground truth tree from any alternative tree by a distance proportional to the loss of the alternative tree.

Figure 3.10 shows the modification to the code of our perceptron, to introduce large margin and root loss.

Input: $D = \{(x, y)\}$ binary data set
r root loss,
C margin,
Output: w
$w \leftarrow 0$
while no convergence
for each $(x, y) \in D$
$\widetilde{y} \leftarrow arg\, max_{h \in H_y(x)}\, s(x, h; w)$
$\widehat{y} \leftarrow arg\, max_{h \in H(x)}\, s(x, h; w) + C * l_r(h, \widetilde{y})$
$w \leftarrow w + \Phi(x, \widetilde{y}) - \Phi(x, \widehat{y})$
return w

Figure 3.10: Latent and Structure Perceptron algorithm with margin and root loss.

3.5.4 Sparse Perceptron

The binarization of the features and the induction of attributes significantly increase the dimension of our features. Many features in high-dimensional space are usually non-informative or noisy. Those noisy features will decrease the generalization performance of a perceptron and derogate from their promising results for dealing with non-linear separable data, especially when they are many noisy features Liu and Tsang (2016). Here we are concerned with model selection: which features should be used to define the prediction score? We want to take advantage of high-dimensional inputs while selecting values of w that achieve high accuracy not only on training data, but also to generalize well on new data. The fact is that models with few features or sparse models are desirable for several reasons like compactness, interpretability, good generalization Martins et al. (2011).

Goldberg and Elhadad (2011) have proposed a modification on the prediction step, inspired in the number of times that a feature participates in models updates. To achieve this goal we include in our perceptron a vector u with the number of updates of w in which every feature has participated. Also, we use a threshold T to establish when a feature became relevant. In other words, T is the lower bound of the number of updates of w required by an attribute to participate in the activation rule of the learning algorithm.

Input: $D = \{(x, y)\}$ binary data set
r root loss,
C margin,
T threshold
Output: w
$w \leftarrow 0$
$u \leftarrow 0$
while no convergence
for each $(x, y) \in D$
$\widetilde{y} \leftarrow arg\, max_{h \in H_y(x)}\, s(x, h; w, u, T)$
$\widehat{y} \leftarrow arg\, max_{h \in H(x)}\, s(x, h; w, u, T) + C * l_r(h, \widetilde{y})$
$w \leftarrow w + \Phi(x, \widetilde{y}) - \Phi(x, \widehat{y})$
$u \leftarrow u + 1[\Phi(x, \widetilde{y}) \neq \Phi(x, \widehat{y})]$
for all s.t $u_i < T$
$u_i \leftarrow 0$
return w

Figure 3.11: Latent, Structure and Sparse Perceptron algorithm with margin and root loss.

Figure 3.11 exhibits the modifications that were made to the perceptron to filter the most used attributes during the training. First, the update of the vector u which also restarted the non-selected features at the end of each epoch. Second, was necessary to modify the scoring function s to include as parameters, the vector u, and the threshold T. The main change in this function occurs in the updating of the weights of each arc in $G(x)$, since we do not consider now those attributes whose corresponding value in the vector u is less than T. The new way of updating the weights is exposed below:

$$w(e) = \sum_{i=1}^{m} \phi_i(x, e) * w[i] * 1[u[i] \geq T]$$

We also introduce another change to the sparse perceptron, since, we modify our update rule for u to not always increase the counter, such us if a feature participates in an update of the current model w. We consider a probability of dropping out this feature, about a given probability *Dropout*. This technique allows better control over the number of selected attributes. This modification is an adaptation for sparse perceptron of the idea of dropout of neural networks, proposed by Hinton et al. (2012); Motta (2014). In Figure 3.12, we present the pseudo-code of the structure perceptron integrating the dropout extension.

With the introduction of dropout extension, we can make a finest selection of attributes, through variations of the *Dropout*. In the original

Input: $D = \{(x, y)\}$ binary data set
r root loss,
C margin,
T threshold
$Dropout$
Output: w
$w \leftarrow 0$
$u \leftarrow 0$
while no convergence
for each $(x, y) \in D$
$\widetilde{y} \leftarrow arg\, max_{h \in H_y(x)}\, s(x, h; w, u, T)$
$\widehat{y} \leftarrow arg\, max_{h \in H(x)}\, s(x, h; w, u, T) + C * l_r(h, \widetilde{y})$
$w \leftarrow w + \Phi(x, \widetilde{y}) - \Phi(x, \widehat{y})$
$u \leftarrow u + 1[\Phi(x, \widetilde{y}) \neq \Phi(x, \widehat{y})\ and\ random[0, 1) < Dropout]$
for all s.t $u_i < T$
$u_i \leftarrow 0$
return w

Figure 3.12: Latent, Structure and Sparse Perceptron algorithm with margin, root loss and dropout.

algorithm, we do not get such most excellent selection of attributes because the threshold of minimum updates is an integer.

3.5.5 Average Perceptron

The last modification we will make to our perceptron concerns the output of the vector of weights w. If we return the most recent version of the weight vector w, intuitively, this version is over-adapted to the last few instances, and may work less well for other instances. For that reason we introduce the technique of average, that return the average of all versions of the weight vector w. We use the averaged structured perceptron, based on the suggested by Collins (2002).

In the classic version of the average perceptron, is stored the sum of all versions of w, plus a counter t of all instances with which the perceptron is trained and finally the output of the perceptron is w/t. We introduced the modification of instead of dividing the sum of all versions of w between the same value for each feature. We store in a vector t the times when the feature ϕ_k affects w, that is when $w[k]$ changes. Then the final result of $w[k]$ is given by dividing the sum of all $w_i[k]$ by $t[k]$. As we will show later, this modification produced better solutions than the classic proposed by Collins (2002) who is used in our reference paper Fernandes et al. (2014).

In Figure 3.13 we show the final version of our perceptron.

Input: $D = \{(x, y)\}$ binary data set
r root loss,
C margin,
T threshold
$Dropout$
Output: w
$w \leftarrow 0$
$u \leftarrow 0$
$t \leftarrow 0$
$s \leftarrow 0$
while no convergence
for each $(x, y) \in D$
$\widetilde{y} \leftarrow arg\, max_{h \in H_y(x)}\, s(x, h; w, u, T)$
$\widehat{y} \leftarrow arg\, max_{h \in H(x)}\, s(x, h; w, u, T) + C * l_r(h, \widetilde{y})$
$w \leftarrow w + \Phi(x, \widetilde{y}) - \Phi(x, \widehat{y})$
$s \leftarrow s + w$
$u \leftarrow u + 1[\Phi(x, \widetilde{y}) \neq \Phi(x, \widehat{y})\ and\ random[0, 1) < Dropout]$
$t \leftarrow t + 1[\Phi(x, \widetilde{y}) \neq \Phi(x, \widehat{y})]$
for all s.t $u_i < T$
$u_i \leftarrow 0$
return $w \leftarrow (s_i/t_i)_{i=0}^{k}$

Figure 3.13: Average, Latent, Structure and Sparse Perceptron algorithm with margin, root loss and dropout.

3.6 Chapter Conclusions

In this chapter, we present the whole process to solve the problem of coreference resolution. Showing the two subtasks in which it is divided: mention detection and mention clustering. To solve the first, we present a predictor based on heuristic rules. The second subtask is much more complicated, and if it requires a machine learning algorithm Throughout the sections in this chapter, we were constructing our predictor by first modifying the data set and exposing the necessary structures to explain our structured and sparse perceptron finally.

4
Empirical Evaluation

Coreference resolution is a NLP task that consists of determining which mentions in a discourse refer to the same entity or event. A mention is a referring expression that has an entity or event as a referent Sapena et al. (2013).

The CoNLL 2012 Shared Tasks have created a standard for: task formalization, data sets and evaluation metrics. This standarization has served to set up a common input ground for the parsers and their further evaluation. In the years following this set up, several machine learning approaches adopt this standard to evaluate their performances.

In the reported experiments, we use a computer with a four processors of 3.2-GHz and 28 GB RAM. regarding training, the feature induction procedure takes 4 hours; loading the training data set and generating features from the learned templates take 10.5 minutes; and the training algorithm takes 9 minutes for epoch. The test procedure on the development set takes 1.3 minutes to load the data set, 5 seconds to load the model, and 10 seconds to predict the output structures for all documents. Most of these procedures could be further optimized in terms of both execution time and memory use.

Throughout this chapter, we explain our results in the different stages of our predictor. Firstly, we will present our final result showing that our predictor has a competitive performance with the state-of-the-art of the problem. Next, we expose the performance in the mention detection subtask. Then we will present results in the creation of the candidate pairs graphs. Finally, we show the results of our coreferences predictor, crossing each of the modifications related to it, such as induction of attributes, margin and loss value, and average.

4.1
CoNLL-2012 Data set

The CoNLL-2012 Shared Task is dedicated to the modeling of unrestricted coreference resolution in three languages: Arabic, Chinese, and English. The OntoNotes project citedoddington2004automatic provide this data sets. We use only the English portion of this data set. In addition to coreference information, the shared task data sets contain various annotation layers, namely, POS, syntactic parses, word senses, NE, and semantic roles (SRL).

The work consists of the automatic identification of coreferring mentions of entities and events, given the predicted information on other layers. The data set for each language comprises three subsets: training, development, and test. We report our system performances on the development and test sets. We obtain the development using models that have been trained only on the training sets. However, we achieve the test results by training on a larger data set obtained by concatenating the training and development sets.

4.2 Evaluation Metrics

We evaluated the system performance using three evaluation metrics: MUC Vilain et al. (1995), B^3 Bagga and Baldwin (1998), and $CEAF_e$ Luo (2005). Each metric presents different strengths and weaknesses. Also, the final value is the average of the F_1 of each of the metrics, as measured in the CoNLL-2012 Shared Task. We will briefly explain each of the metrics below.

MUC metrics evaluated the set of system clusters by looking at the minimum number of pair additions and removals required for them to match the ground truth clusters Vilain et al. (1995). The pairs to be added represented false negatives, while the pairs to be removed represented false positives. Let K represent the ground truth clusters set, and R the system clusters set. Given clusters k and r from K and R, respectively, MUC recall and precision of R were:

$$recall = \frac{\sum_k(|k| - m(k,R))}{\sum_k(|k| - 1)}$$

$$precision = \frac{\sum_k(|r| - m(k,K))}{\sum_k(|r| - 1)}$$

where $m(r,K)$ is the number of clusters in K that intersected the cluster r.

B^3 metrics evaluated system performance by measuring the overlap between the clusters predicted by the system and the ground truth clusters Bagga and Baldwin (1998). Let C be a collection of N documents, d a document in C, and m a mention in document d. We defined the ground truth cluster that included m as G_m and the system cluster that contained m as S_m. O_m was the intersection of G_m and S_m. B^3 recall and precision were defined as:

$$recall = \frac{1}{N}\sum_{d\in C}\sum_{m\in d}\frac{|O_m|}{|G_m|}$$

$$precision = \frac{1}{N}\sum_{d\in C}\sum_{m\in d}\frac{|O_m|}{|S_m|}$$

$CEAF_e$ uses a similarity measure ϕ_4 to evaluate the similarity of two

entities. It uses the Kuhn-Munkres Kuhn (1955); Munkres (1957) algorithm to find the best one-to-one mapping of the cluster to the response entities (g^*) using the given similarity measure. Luo (2005). Let gold standard clusters in a document d be $K = \{k_i : i = 1, 2, ..., |K|\}$, and system clusters in a document d be $R = \{r_i : i = 1, 2., |R|\}$. The chain-based ϕ_4 scores were defined as:

$$\phi_4(k_i, r_j) = \frac{2 * |k_i \cap r_j|}{|k_i| + |r_j|}$$

The $CEAF_e$ precision and recall were defined as:

$$recall = \frac{\sum_{k_i \in K} \phi_4(k_i, g^*(k_i))}{\sum_{k_i \in K} \phi_4(k_i, k_i)}$$

$$precision = \frac{\sum_{k_i \in K} \phi_4(k_i, g^*(k_i))}{\sum_{r_i \in R} \phi_4(r_i, r_i)}$$

For the three metrics the value of F_1 is given by the equation.

$$F_1 = \frac{2 * recall * precision}{recall + precision}$$

To compare our systems with the rest of the systems participating in the CoNLL-2012 Shared Task we use the official scorer utilized in the competition.

4.3 State-of-the-Art Systems

System	CoNLL
Clark and Manning (2016)	65.73
Wiseman et al. (2016)	64.21
Fernandes et al. (2014)	63.37
This Work	**62.24**
Martschat et al. (2012)	61.31
Björkelund and Farkas (2012)	61.24
Chang et al. (2012)	60.18

Table 4.1: State-of-the-art systems.

In Table 4.1, we present the CoNLL scores of the best-performing systems on the coreference resolution problem. The first two rows are very recent systems that use modern techniques such as deep learning, this is the reason for the increase in the score of these systems. The third best system is the one we used as the basis for our work, the result shown is the same that they competed in the CoNLL-2012 Shared Task Fernandes (2012) because in Fernandes et al. (2014) they modify the results only for the Chinese language.

Our system has the fourth best result, and in the CoNLL 2012 competition, would have obtained the second place for the English language. The last three rows of this table correspond to the competitors that are ranked the second Martschat et al. (2012), third Björkelund and Farkas (2012), and fourth Chang et al. (2012) in the shared task for English.

Table 4.2 shows in detail the results of the best systems in the state-of-the-art. Note that our system, for the metric B^3, has the worst F_1 of all the systems that participated in the competition, but it compensates having much better F_1 than the systems of the last three rows in the others two metrics.

System	*MUC*			B^3			$CEAF_e$			CoNLL
	R	P	F_1	R	P	F_1	R	P	F_1	
Clark and Manning (2016)	79.19	70.44	74.56	69.93	57.99	63.40	63.46	55.52	59.23	65.73
Wiseman et al. (2016)	77.49	69.75	73.42	66.83	56.95	61.50	62.14	53.85	57.70	64.21
Fernandes et al. (2014)	65.83	75.91	70.51	65.79	77.69	71.24	55.00	43.17	48.37	63.37
This Work	**65.67**	**73.15**	**69.21**	**64.71**	**74.58**	**69.3**	**53.17**	**44.09**	**48.21**	**62.24**
Martschat et al. (2012)	65.21	68.83	66.97	66.50	74.69	70.36	48.64	44.72	46.60	61.31
Björkelund and Farkas (2012)	65.23	70.10	67.58	65.90	75.24	70.26	48.60	43.42	45.87	61.24
Chang et al. (2012)	64.77	68.06	66.38	67.04	71.81	69.34	46.64	43.11	44.81	60.18

Table 4.2: State-of-the-art systems details.

4.4 Mention Detection

The first subtask to solve in our system is the mentions detection. This task is of vital importance in the final result because those mentions that are not detected in this first stage will be lost throughout the process. Then it is important first to ensure the recall meets the precision.

In Table 4.3, we present these performances in the development and training sets. Indeed a great difference between recall and precision can be seen. The *Total* column indicates how many mentions are annotated in the gold standard data set, which comprises non-singletons only. The *Extracted* column indicates the number of mentions detected by our system, and the *Correct* column indicates how many of them are correct. The columns R, P, and F_1 correspond, respectively, to the recall, precision, and F_1-score of mentions.

Data set	Total	Extracted	Correct	R	P	F_1
development	19 155	44 322	18 021	94.08	40.66	56.78
train	155 141	352 572	146 859	94.66	41.65	57.85

Table 4.3: Mention detection performances before clustering.

As in Fernandes et al. (2014) in our system, we remove singleton mentions only after the mention clustering step. Hence, for each model, we assess mention

detection twice: before and after the mention clustering step. In Table 4.4 we show the results obtained in the mentions detection for the development set, after clustering them. We only add a column to this table in comparison to Table 4.3, and it is *EFI* that represents whether the results are using the features induction or not.

Total	EFI	Extracted	Correct	R	P	F_1
19 155	No	16 927	12 995	67.84	76.77	72.03
	Yes	17 205	14 155	73.89	82.27	77.86

Table 4.4: Mention detection performances on development set after clustering.

Note that after the mention clustering step, the mention detection precision increases because many correctly identified singletons are excluded. Of course, some non-singleton mentions are wrongly identified as singletons in the mention clustering step, and there is a consequent drop in recall.

4.5 Candidate Pair Generation

The candidate pair generation step is responsible for creating the candidate pairs graph G(x) given the set of detected mentions. Our approach to this subtask is based on the sieves used in Fernandes et al. (2014), proposed by Lee et al. (2013). We add to the sieves the precondition that one mention is not embedded in another. We introduced this filter to achieve a better adaptation to the English portion of the CoNLL-2012 Shared Task data sets.. The purpose of using sieves in this step is twofold. First, when generating G(x), we want to include only the arcs that link mentions that are likely to be coreferent to avoid dealing with a dense graph in the subsequent steps, which involve arc feature generation and solving maximum branching problems on G(x). Second, we do not want to generate too many precision errors for the subsequent steps by linking mentions that are not likely to be coreferent. Conversely, the sieves cannot be too restrictive, or they would miss too many intracluster arcs and put coreferent mentions in different connected components. If two coreferent nodes m1 and m2 belong to different connected components in G(x), the coreference tree predictor is unable to generate a tree on G(x) containing the coreferent nodes m1 and m2. Therefore, there is a trade-off between the restrictiveness of the sieves and the coreference resolution recall.

In this section, we will show the recall of our candidate arcs, evaluated in the metrics MUC, B^3 and $CEAF_e$. For this we create the candidate graphs $G(x)$ for each document and pass as predictor input the graph $G_y(x)$. In this

way, we will evaluate the recall of corrects arcs, that is, those that connect coreferent mentions.

Table 4.5 report the impact of different combinations of sieves on the MUC, B^3 and $CEAF_e$ recall of the candidate arcs. These results help us to understand the contribution of each sieve and the overlap among them. The first column shows some combinations of sieves. The *Arcs* column indicates the number of arcs generated, that is, the sum of the arcs of $G(x)$ for each document. In the next column, we show the number of correct arcs, which is the sum of the arcs of each graph $G_y(x)$ for each document. The next three columns show the recall in each of the metrics when we pass $G_y(x)$ as the input of our predictor. Finally, the last column shows the average recall of each sieves combination.

Sieves	Arcs	Correct Arcs	Recall			Average
			MUC	B^3	$CEAF_e$	
None	4 488 163	95 027	92.67	93.22	94.53	93.47
(1)	317 532	15 484	62.68	53.66	77.78	64.71
(2)	9 937	5 216	10.88	29.37	57.46	32.57
(3)	90 154	34 579	59.64	51.04	74.62	61.77
(4)	31 755	19 744	15.64	31.35	54.85	33.95
(5)	240 544	25 170	27.22	35.48	59.67	40.79
(1) and (3)	396 450	44 283	83.92	81.03	89.2	84.72
(1), (3) and (5)	408 478	50 248	84.27	81.76	89.43	85.15
(1), (3), (5) and (4)	594 876	53 621	84.55	82.13	89.51	85.4
(1), (3), (5), (4) and (2)	595 767	53 895	84.96	82.09	89.66	85.57

Table 4.5: Performances of sieves on the development set.

It is important to note in the Table 4.5 that, by placing all the arcs, that is, without any sieve, we achieved an average recall of 93.47, which shows the excellent results of our mentions predictor. Also, using all the sieves, we managed to reduce the number of arcs to 13.27% of the total, obtaining an average recall of 91.55% of the average recall using all the arcs.

4.6 Coreference Resolution

To solve the problem of coreference solution, we use a structured perceptron, to which we perform a group of modifications that allowed to improve our results for this task considerably.

In this section, we will show the results of our predictor, exposing the performance for each of the modifications made to our structured perceptron.

4.6.1 Margin and Loss Value

The first modification we evaluate is the margin and the loss value. These parameters allow us to modify the balance between precision and recall.

In Figure 4.1, we show the impact on the CoNLL scorer provided by the margin and loss value. We report the per-epoch performances of the current model on the development set during the first 50 epochs. Each epoch corresponds to iterating over all instances in the training data set. First, we train our perceptron use neither margin or loss value. Second, we specialize the perceptron by using only margin with the parameter $C = 1,000$. And finally, we examine the large margin averaged perceptron with the loss value $r = 0.5$. The values given to the parameters C and r were calibrated during the experiments and were used to train our best model. Using the margin, we observe a consistent improvement in all epochs, and in the final model using both, margin and loss value, we obtain a significant improvement of more than 1%.

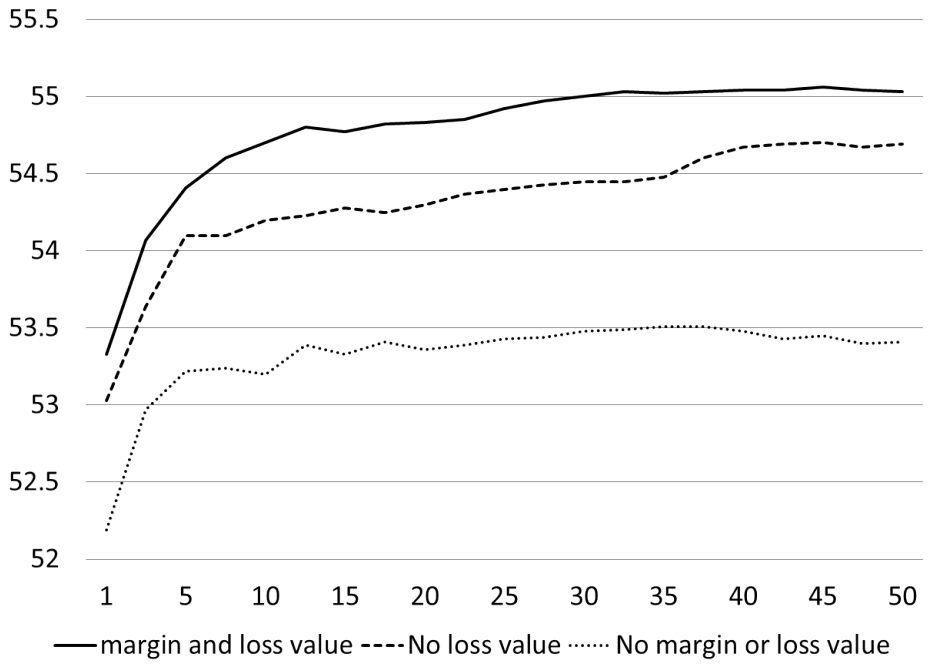

Figure 4.1: Impact of margin and loss value on performance.

Additionally, we see that our method seems to converge before 50 epochs, which is the number of epochs used to train all models used in this work.

4.6.2 Average

In our system, we use an average perceptron, because the averaged algorithm is more robust to noisy examples and usually performs significantly better than the non-averaged version. But we proposed change in the way we do the average.

The Figure 4.2 shows the performance in the CoNLL score of the development set, without induction of attributes, using the two average strategies explained in section 3.5 and without average. It can be observed how when not used average the results are very fluctuating, varies a lot between one epoch and another. Also, we can appreciate how using the traditional average strategy the results are stable, generally increasing. Finally using our the average technique, that we proposed in this work, we were able to improve the performance of our algorithm by 0.5%.

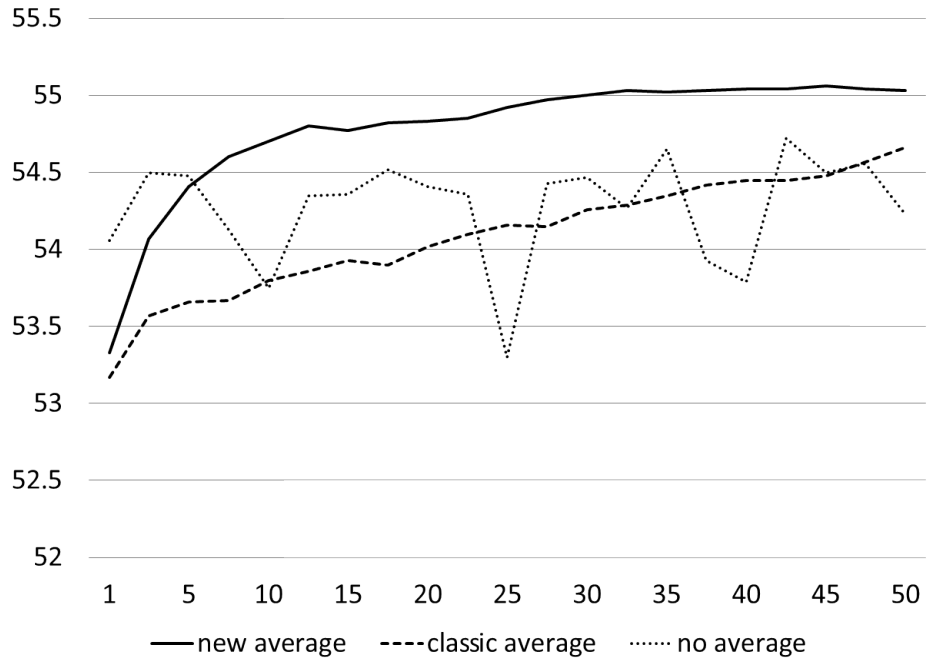

Figure 4.2: Impact of average on performance.

4.6.3 Feature Induction

Using EFI, we can automatically generate feature templates without human effort, which allows us to experiment with different template sets easily. In our experiments, we use all the basic features to produce feature templates from them. Only 64 basic features appear at least once in the template set. In

Table 4.6 we present the number of appearances that have the features selected in the templates.

Feature	Appearances
L2	217
Sy12i	128
Se7	92
L1i	79
Sy3j	68
L15	60
L3	51
Se1	43
P4	37
Se3j	32
L11j	32
L6jr1	25
L1j	21
L12	20
Se8	17
Sy4j	16
Sy1j	16
Sy10	15
Sy11j	15
L11i	15
P3	15
Sy7j	13
Sy9ir	13
Sy11i	11
Sy9jr	10
L8	9
L7j	8
Se4	8
L10i	7
L6jl2	7
L6il2	7
L5	6
Se2j	6
L10j	6
Sy9jl	6
L6il1	5

Continuation of Table 4.6

Feature	Appearances
Sy4i	5
Sy6	5
Se6	5
L6ir1	5
L4	5
Sy2jl1	4
Sy12j	4
Se5jr	4
Sy5	4
Se5jl	4
Sy2jr1	4
Se3i	4
Sy1i	3
L6ir2	3
L6jr2	3
Sy2jr2	3
Sy2jl2	3
Se2i	3
L7i	3
Sy2ir1	3
Sy13i	3
Sy7i	3
L6jl1	3
Sy13j	2
Sy9il	2
Se5il	2
Sy2il1	2
Se5ir	2

Table 4.6: Appearances of the features in the templates.

In our final system, we use a set of 252 templates obtained by merging the output of two independent EFI executions. These two runs are based on two different training data sets. In the first we use only the mention pairs produced by sieves 1 and 3; and in the second the mention pairs produced by all sieves. In order not to generate too many templates we limit the height of the decision tree to 5 and create the templates in sizes from 2 to 5.

In Table 4.7 we report the performance of our system evaluated in the development set. In the first row, the results are observed using only the 73 basic features. While for the results shown in the second row we also use the features generated by the 252 templates. The results are impressively better since the value of F_1 for each metric increases by more than 5%, with the metric MUC being the one with the highest increase of 9.39% more. The final result grew by 7.25%.

EFI	MUC			B^3			$CEAF_e$			CoNLL
	R	P	F_1	R	P	F_1	R	P	F_1	
No	55.63	62.86	59.02	59.47	69.57	64.12	46.12	38.49	41.96	55.03
Yes	64.81	72.43	68.41	65.59	75.01	69.99	53.34	44.36	48.44	62.28

Table 4.7: Impact of EFI in development performance.

4.6.4 Sparse Perceptron

Due to the induction of attributes in our system, the number of features grows considerably in the order of millions. Then it is necessary to reduce the size of the model. For this, we introduce regularization techniques based on sparse linear models. Our primary goal is to promote more compact models, which are desirable because they facilitate interpretation, they take less time to be trained and occupy less computational resources.

Table 4.8 presents the performance on the CoNLL score obtained in the development set after 15 training epoch, using induction of attributes. This table presents the impact of the threshold and dropout parameters on the number of attributes of the generated model, and also shows a slight improvement in performance. The columns *Threshold* and *Dropout* show the values of the parameters used. The *#Features* column exposes the number of attributes to use in the resulting model.

Threshold	Dropout	# Features	CoNLL
0	1	4 463 179	61.74
5	1	458 624	61.72
5	0.9	431 552	61.74
5	0.8	403 013	61.71
5	0.6	333 138	61.74
5	0.4	244 719	61.6

Table 4.8: Impact of threshold and dropout in the model length and development performance.

It is important to note how the number of attributes in the models decreases when we enter threshold and dropout to our perceptron, all without

reducing the performance, as in cases with dropout 0.9 and 0.6, on the contrary when we increase the number of epochs in training the performance of our system improves.

In the scenario of feature induction the amount of binary attributes that are generated by the templates can lead to a potential overfitting of the training data. Also, it might introduce a large number of irrelevant attributes as was shown in Motta (2014).

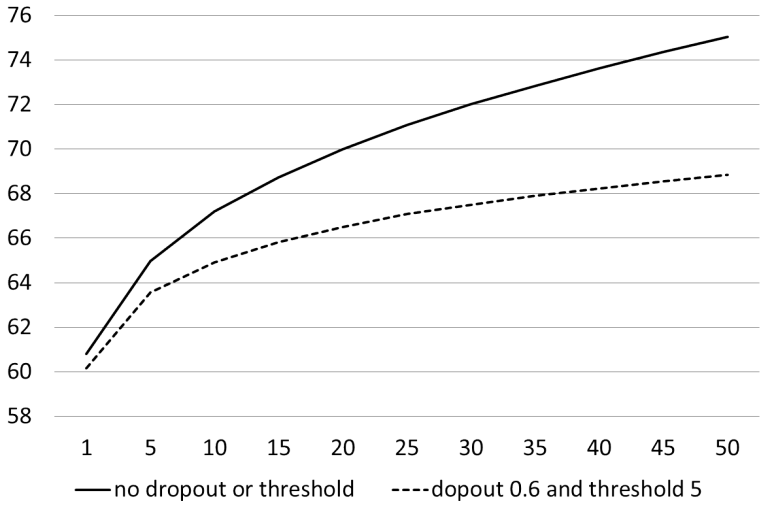

Figure 4.3: Impact of dropout and threshold on training performance.

Figure 4.3 and Figure 4.4 confirm the previous proposition of Motta (2014), as, Figure 4.3 shows how the performance during training was reduced when we introduce the sparse perceptron technique. Although Figure 4.4 exposes how performance increases in the development set.

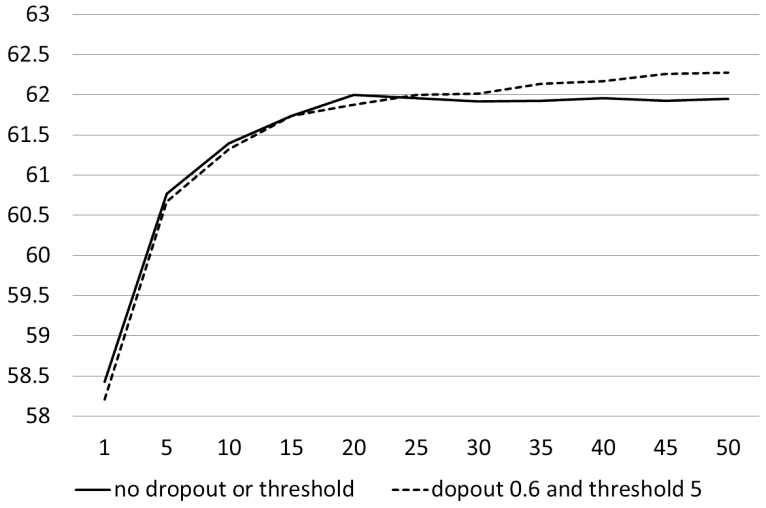

Figure 4.4: Impact of dropout and threshold on development performance.

4.7 Error Analysis

In this section, we provide statistics regarding the most common errors within the candidate pairs generation and the outputs of our structure predictor for the development set. Three steps in our predictor are direct sources of errors: mention detection, candidate pair generation, and mention clustering. The errors in each step propagate to the next steps.

During the detection of mentions, we focus more on the recall than on precision. One of the errors during this stage is the non-detection of the singleton mentions since we detect this type of mentions only during the prediction. Also, our system does not handle coreferring mentions of events, and it thus does not consider verbs when creating candidate mentions, which is responsible for many recall errors in this step.

In the candidate pairs generation stage, we use sieves to choose the edges that are part of $G(x)$. This method can select many incorrect edges to be part of the candidate pair graph, as well as leave out arcs between corefents mentions. The three first columns in Table 4.9, shows the head POS tags combinations, of edges (i, j) incorrectly added to $G(x)$ that occur most frequently during the creation of pairs graph candidates in development set. The remaining columns show the head POS tag combinations of the (i, j) edges that join coreferents mentions, but were not added to the candidate pairs graph. The POS tags that appear in the table are: NNP and NR stand for singular proper noun, NN for singular noun, NNS for plural noun, NT for temporal noun, PRP for pronoun, and PRP$ for possessive pronoun.

Wrong Selected			Not Selected		
i	j	%	i	j	%
PRP	PRP	19.11	PRP	PRP	22.55
NN	PRP	15.99	NNP	PRP	11.92
NN	NN	9.1	PRP	NNP	11.58
NNS	PRP	6.42	PRP$	PRP	9.86
NNP	NNP	5.36	PRP	PRP$	8.88
NNP	NN	4.84	NNP	PRP$	4.31
NN	NNP	4.13	PRP$	NNP	3.88
NNP	PRP	4.13	NN	PRP	3.48
PRP	NN	3.77	NNP	NNP	3
NNS	NN	3.33	PRP	NN	2.87
Others		23.82	Others		17.67

Table 4.9: Candidate pairs generation errors.

Note, that the major problems in both cases are obtained when the head of two mentions are pronouns, which makes to suspect that the significant part

of the errors in this stage come from the sieve (5). That, added to the fact that this sieve scores a huge number of arches without obtaining a significant recall, in comparison with the rest of the sieves, indicates that for future work this sieve must be revised.

To measure error during training, we compare each predicted document tree $\widehat{y}$ with the corresponding constrained document tree $\widetilde{y}$. For each incorrectly predicted parent i for a mention j, that is i and j are not in the same cluster, let $\widehat{i}$, be its corresponding correct arc in $\widetilde{y}$. For all documents in the training set, we compute the frequency of the head POS tags of mentions j, i, and $\widehat{i}$, or *Root*, in case that i or $\widehat{i}$ was the root node. Note that a constrained document tree is computed on the graph generated in the candidate pair generation step, which already includes recall errors. Thus, the frequencies reported here are computed over the remaining errors. In Table 4.10, we present the top ten errors, after one training epoch, where, again, the greatest number of mistakes occurs between mentions with pronoun head POS.

i	$\widehat{i}$	j	%
PRP	PRP	PRP	12,39
Root	NNP	NNP	9,58
NNP	Root	NNP	9,02
Root	NN	NN	6,63
NN	Root	NN	6,48
PRP	Root	PRP	5,78
Root	PRP	PRP	5,66
NNP	NNP	NNP	4,88
NNS	Root	NNS	3,02
Root	NNS	NNS	1,92
Others			35.18

Table 4.10: Most frequent errors whenever an incorrect parent i for a mention j is predicted instead of the correct parent $\widehat{i}$.

Finally, we present the error analysis during the prediction stage. A highly probable error of our predictor is to connect a singleton mention to a document tree. In Table 4.11, we present the most frequent head POS tags for these singleton errors, surprisingly, most errors are nouns.

i	%
NN	47,44
NNS	20,97
NNP	19,59
PRP	7,84
Others	4.16

Table 4.11: Most frequent singleton errors.

4.8 Chapter Conclusions

Here we reported the empirical evaluation performed to the proposed solution. We have observed our competitive results when compared against state-of-art approaches. We also provide detailed experimental results that highlight the contribution of individual parts of our system, providing important insights to researchers interested in coreference resolution. Finally, we did an analysis of the most common errors that affect the performance of our system.

5
Conclusion

Several techniques have been proposed to automatically solve the coreference resolution problem. The CoNLL 2011 and 2012 Shared Tasks, marked an important point in the advances to address the issue of coreference resolution, because, in addition to providing an excellent annotated data set, they establish a standard for evaluating and comparing parsers. In this dissertation, we examine the most relevant works on this topic, showing the different types of approaches to the problem, among which the most used are mention-pair classification, entity-mention, correlation clustering and more recently, deep learning has been used to address the problem.

Following a line of research of *Laboratorio de Engenharia de Algoritmos e Redes Neurais* (LEARN), we base our work on the paper Fernandes et al. (2014), that is based on two modeling techniques: latent coreference trees and entropy-guided feature induction. We represent a document using a graph, in which the nodes are the mentions detected in the document. The arcs of this graph link mention pairs that are coreferent candidates. Our learning algorithm learns to extract from each graph a tree that provides the best clustering.

In this dissertation, we use the structured perceptron as our learning algorithm, because this algorithm is one of the most popular linear discriminant methods due to its flexibility, robustness, and simplicity. It is also adapted to learn complex representations, such as those required for tree prediction, as in our case. We make modifications like the introduction of dropout and sparse vector, to reduce the number of parameters of our algorithm. We modify the classic average technique used in Fernandes et al. (2014), improving the performance of our algorithm by 0.5%. Also we include an induction method, which automatically provides features that represent nonlinear patterns.

With the inclusion of the techniques of dropout and sparse perceptron, we managed to reduce the number of attributes of the model in 92,54%, which reduced the overfitting in our algorithm, since, it decreases the performance on the training set but increases it on the development set. Also, our solution significantly reduces the time to obtain the clusters of a document, since, for the prediction, our system takes 0.35 seconds per document, while in state-of-the-art Clark and Manning (2016), it takes 5 seconds for each one.

Our coreference resolution system for the English language system obtained a performance of 62.24 on the CoNLL score in the test set. As far as we

know, this result is the fourth best result found in the state-of-the-art systems, and in the CoNLL 2012 Shared Task would have obtained the second place for the English language.

Future Work

Our error analysis shows that most of the errors during the candidate pairs graph generation and training, come from edges erroneously created between pronouns. In a future work, it is necessary to work on the pronouns resolution.

In our system the set of attributes in their vast majority are extracted from Fernandes et al. (2014), so they are general attributes that can be used in any language. For more specification in the English language, it is necessary to embed more specific attributes, such as those using Wordnet Miller (1995) proposed in Santos and Carvalho (2011).

One limitation of the proposed modeling approach is the arc-based features. Such local information is clearly not sufficient to model all dependencies involved in coreference resolution. Then, considering more sophisticated contextual features is necessary. Hence, in future work, we plan to include higher-order features. Higher-order tree-based features have been successfully applied to dependency parsers McDonald and Pereira (2006); Koo et al. (2010) and can be used to extend our coreference system.

6
Bibliography

BAGGA, A.; BALDWIN, B. Algorithms for scoring coreference chains. In **The first international conference on language resources and evaluation workshop on linguistics coreference**. [S.l.: s.n.], 1998. vol. 1, p. 563–566. 1.1, 4.2

BANSAL, M.; KLEIN, D. Coreference semantics from web features. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1**. [S.l.], 2012. p. 389–398. 2.1

BENGTSON, E.; ROTH, D. Understanding the value of features for coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the Conference on Empirical Methods in Natural Language Processing**. [S.l.], 2008. p. 294–303. 2.1, 2.2

BERGSMA, S.; LIN, D. Bootstrapping path-based pronoun resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics**. [S.l.], 2006. p. 33–40. 5

BJÖRKELUND, A.; FARKAS, R. Data-driven multilingual coreference resolution using resolver stacking. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Joint Conference on EMNLP and CoNLL-Shared Task**. [S.l.], 2012. p. 49–55. 4.3, 4.3

CAI, J.; MÚJDRICZA-MAYDT, E.; STRUBE, M. Unrestricted coreference resolution via global hypergraph partitioning. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task**. [S.l.], 2011. p. 56–60. 2.7

CAI, J.; STRUBE, M. End-to-end coreference resolution via hypergraph partitioning. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 23rd International Conference on Computational Linguistics**. [S.l.], 2010. p. 143–151. 2.7

CHANG, K.-W. et al. Inference protocols for coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task**. [S.l.], 2011. p. 40–44. 2.1

CHANG, K.-W. et al. Illinois-coref: The ui system in the conll-2012 shared task. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Joint Conference on EMNLP and CoNLL-Shared Task**. [S.l.], 2012. p. 113–117. 4.3, 4.3

CHU, Y.-J.; LIU, T.-H. On shortest arborescence of a directed graph. **Scientia Sinica**, SCIENCE PRESS 16 DONGHUANGCHENGGEN NORTH ST, BEIJING 100717, PEOPLES R CHINA, vol. 14, no. 10, p. 1396, 1965. 2.5, 3.5

CLARK, K.; MANNING, C. D. Deep reinforcement learning for mention-ranking coreference models. **arXiv preprint arXiv:1609.08667**, 2016. 1.3, 4.3, 4.3, 5

CLARK, K.; MANNING, C. D. Improving coreference resolution by learning entity-level distributed representations. **arXiv preprint arXiv:1606.01323**, 2016. 2.6

COLLINS, M. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10**. [S.l.], 2002. p. 1–8. 2.4, 3.5.1, 3.5.5

DENIS, P.; BALDRIDGE, J. et al. Joint determination of anaphoricity and coreference resolution using integer programming. In CITESEER. **HLT-NAACL**. [S.l.], 2007. p. 236–243. 2.7

EDMONDS, J. Optimum branchings. **Journal of Research of the national Bureau of Standards B**, vol. 71, no. 4, p. 233–240, 1967. 2.5

FERNANDES, E. L. R. **Entropy guided feature generation for structure learning**. Tese (Doutorado) — PUC-Rio, 2012. 4, 3, 1.3, 3.4, 3.5, 3.5.3, 4.3

FERNANDES, E. R.; BREFELD, U. Learning from partially annotated sequences. In SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. [S.l.], 2011. p. 407–422. 3.5.1

FERNANDES, E. R.; SANTOS, C. N. dos; MILIDIÚ, R. L. Latent trees for coreference resolution. **Computational Linguistics**, MIT Press, 2014. 1, 1.1, 1.1, 1.2, 2, 1.3, 2, 2.5, 3, 3.2, 3.2, 3.3, 3.4, 3.5, 3.5.1, 3.5.3, 3.5.5, 4.3, 4.3, 4.4, 4.5, 5

FINKEL, J. R.; MANNING, C. D. Enforcing transitivity in coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers**. [S.l.], 2008. p. 45–48. 2.7

FINLEY, T.; JOACHIMS, T. Supervised clustering with support vector machines. In ACM. **Proceedings of the 22nd international conference on Machine learning**. [S.l.], 2005. p. 217–224. 2.4

GOLDBERG, Y.; ELHADAD, M. **Learning sparser perceptron models**. [S.l.], 2011. 4, 3.5.4

GRISHMAN, R.; SUNDHEIM, B. Message understanding conference-6: A brief history. In **Coling**. [S.l.: s.n.], 1996. vol. 96, p. 466–471. 1.1

HAGHIGHI, A.; KLEIN, D. Coreference resolution in a modular, entity-centered model. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics**. [S.l.], 2010. p. 385–393. 2.1

HARABAGIU, S. M.; BUNESCU, R. C.; MAIORANO, S. J. Text and knowledge mining for coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies**. [S.l.], 2001. p. 1–8. 2.2

HINTON, G. E. et al. Improving neural networks by preventing co-adaptation of feature detectors. **arXiv preprint arXiv:1207.0580**, 2012. 3.5.4

KIM, J.-D. et al. Genia corpus—a semantically annotated corpus for bio-textmining. **Bioinformatics**, Oxford Univ Press, vol. 19, no. suppl 1, p. i180–i182, 2003. 2.3

KLENNER, M. Enforcing consistency on coreference sets. In **Recent Advances in Natural Language Processing (RANLP)**. [S.l.: s.n.], 2007. p. 323–328. 2.7

KOO, T. et al. Dual decomposition for parsing with non-projective head automata. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing**. [S.l.], 2010. p. 1288–1298. 5

KRUSKAL, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. **Proceedings of the American Mathematical society**, JSTOR, vol. 7, no. 1, p. 48–50, 1956. 2.5

KUHN, H. W. The hungarian method for the assignment problem. **Naval research logistics quarterly**, Wiley Online Library, vol. 2, no. 1-2, p. 83–97, 1955. 4.2

LEE, H. et al. Deterministic coreference resolution based on entity-centric, precision-ranked rules. **Computational Linguistics**, MIT Press, vol. 39, no. 4, p. 885–916, 2013. 2, 2.1, 3, 3.2, 4, 4.5

LIU, W.; TSANG, I. W. Sparse perceptron decision tree for millions of dimensions. In **AAAI**. [S.l.: s.n.], 2016. p. 1881–1887. 3.5.4

LUO, X. On coreference resolution performance metrics. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing**. [S.l.], 2005. p. 25–32. 1.1, 4.2

LUO, X. et al. A mention-synchronous coreference resolution algorithm based on the bell tree. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics**. [S.l.], 2004. p. 135. 2.3

MARTINS, A. F. et al. Structured sparsity in structured prediction. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the Conference on Empirical Methods in Natural Language Processing**. [S.l.], 2011. p. 1500–1511. 3.5.4

MARTSCHAT, S. et al. A multigraph model for coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Joint Conference on EMNLP and CoNLL-Shared Task**. [S.l.], 2012. p. 100–106. 4.3, 4.3

MCCALLUM, A.; WELLNER, B. Conditional models of identity uncertainty with application to noun coreference. In **NIPS**. [S.l.: s.n.], 2004. p. 905–912. 2.4

MCCARTHY, J. F.; LEHNERT, W. G. Using decision trees for coreference resolution. **arXiv preprint cmp-lg/9505043**, 1995. 2.2

MCDONALD, R.; CRAMMER, K.; PEREIRA, F. Online large-margin training of dependency parsers. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 43rd annual meeting on association for computational linguistics**. [S.l.], 2005. p. 91–98. 3.5

MCDONALD, R. T.; PEREIRA, F. C. Online learning of approximate dependency parsing algorithms. In **EACL**. [S.l.: s.n.], 2006. p. 81–88. 5

MILIDIÚ, R. L. et al. Phrase chunking using entropy guided transformation learning. In **ACL**. [S.l.: s.n.], 2008. p. 647–655. 4, 3

MILLER, G. A. Wordnet: a lexical database for english. **Communications of the ACM**, ACM, vol. 38, no. 11, p. 39–41, 1995. 2.2, 5

MOTTA, E. N. **Induçao e Seleçao Incrementais de Atributos no Aprendizado Supervisionado**. Tese (Doutorado) — PUC-Rio, 2014. 3.5.4, 4.6.4, 4.6.4

MUNKRES, J. Algorithms for the assignment and transportation problems. **Journal of the society for industrial and applied mathematics**, SIAM, vol. 5, no. 1, p. 32–38, 1957. 4.2

NG, V. Shallow semantics for coreference resolution. In **IJcAI**. [S.l.: s.n.], 2007. vol. 2007, p. 1689–1694. 2.2

NG, V.; CARDIE, C. Combining sample selection and error-driven pruning for machine learning of coreference rules. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10**. [S.l.], 2002. p. 55–62. 2.2

NG, V.; CARDIE, C. Improving machine learning approaches to coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 40th annual meeting on association for computational linguistics**. [S.l.], 2002. p. 104–111. 2.2, 3.3, 3.3

PRADHAN, S. et al. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Joint Conference on EMNLP and CoNLL-Shared Task**. [S.l.], 2012. p. 1–40. 1.1, 1.1, 5, 1.3

PRADHAN, S. et al. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task**. [S.l.], 2011. p. 1–27. 1.1

QUINLAN, J. R. **C45**. [S.l.]: Morgan Kaufmann, 1992. 3.4.2

RECASENS, M. et al. Semeval-2010 task 1: Coreference resolution in multiple languages. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 5th International Workshop on Semantic Evaluation**. [S.l.], 2010. p. 1–8. 1.1

SANTOS, C. N. dos; CARVALHO, D. L. Rule and tree ensembles for unrestricted coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task**. [S.l.], 2011. p. 51–55. 1, 3, 1, 2.1, 3.1, 3.3, 3.3, 3.4, 5

SANTOS, C. N. dos; MILIDIÚ, R. L. Entropy guided transformation learning. In **Foundations of Computational, Intelligence Volume 1**. [S.l.]: Springer, 2009. p. 159–184. 3, 3.4

SAPENA, E.; PADRÓ, L.; TURMO, J. Relaxcor: A global relaxation labeling approach to coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 5th International Workshop on Semantic Evaluation**. [S.l.], 2010. p. 88–91. 3.3

SAPENA, E.; PADRÓ, L.; TURMO, J. A constraint-based hypergraph partitioning approach to coreference resolution. **Computational Linguistics**, MIT Press, vol. 39, no. 4, p. 847–884, 2013. 2.1, 2.7, 4

SOON, W. M.; NG, H. T.; LIM, D. C. Y. A machine learning approach to coreference resolution of noun phrases. **Computational linguistics**, MIT Press, vol. 27, no. 4, p. 521–544, 2001. 2.1, 2.2

STOYANOV, V. et al. Coreference resolution with reconcile. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the ACL 2010 Conference Short Papers**. [S.l.], 2010. p. 156–161. 2.1

SUN, X. et al. Latent variable perceptron algorithm for structured classification. In **IJCAI**. [S.l.: s.n.], 2009. vol. 9, p. 1236–1242. 3.5.1

TSOCHANTARIDIS, I. et al. Large margin methods for structured and interdependent output variables. **Journal of machine learning research**, vol. 6, no. Sep, p. 1453–1484, 2005. 3.5.3

URYUPINA, O. Linguistically motivated sample selection for coreference resolution. In **Proceedings of DAARC**. [S.l.: s.n.], 2004. vol. 2004. 2.2

VILAIN, M. et al. A model-theoretic coreference scoring scheme. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 6th conference on Message understanding**. [S.l.], 1995. p. 45–52. 1.1, 4.2

WISEMAN, S.; RUSH, A. M.; SHIEBER, S. M. Learning global features for coreference resolution. **arXiv preprint arXiv:1604.03035**, 2016. 4.3, 4.3

WISEMAN, S. J. et al. Learning anaphoricity and antecedent ranking features for coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. [S.l.], 2015. 2.6

YANG, X. et al. An entity-mention model for coreference resolution with inductive logic programming. In **ACL**. [S.l.: s.n.], 2008. p. 843–851. 2.3

YANG, X. et al. Coreference resolution using competition learning approach. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1**. [S.l.], 2003. p. 176–183. 2.2

YANGY, X. et al. An np-cluster based approach to coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 20th international conference on Computational Linguistics**. [S.l.], 2004. p. 226. 2.3

YU, C.-N. J.; JOACHIMS, T. Learning structural svms with latent variables. In ACM. **Proceedings of the 26th annual international conference on machine learning**. [S.l.], 2009. p. 1169–1176. 2.4, 2.5, 3.5.1

YUAN, B. et al. A mixed deterministic model for coreference resolution. In ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Joint Conference on EMNLP and CoNLL-Shared Task**. [S.l.], 2012. p. 76–82. 2.1

ZHEKOVA, D.; KÜBLER, S. Machine learning for mention head detection in multilingual coreference resolution. In **RANLP**. [S.l.: s.n.], 2013. p. 747–754. 3.1.1