Statistical coreference resolving in a multi-language domain

Marcus Stamborg
cid03mst@student.lu.se
Lund University - Department of Computer Science

December 21, 2012

.

Abstract

In many applications it is interesting to link names and expressions to real world entities. Since there are many ways in which to refer to the same entity, a system not being able to link such mentions will perform poorly. The process of coreference resolving is a method to handle the process of linking mentions that refer to the same entity together.

This Master's thesis presents the additions created and added to a pre-existing co-reference resolving system used in the Conll-2011 (Pradhan et al., 2011) shared task. The current system builds upon the previous system by Björkelund and Nugues (2011) and describes the improvements made, mainly concerning non-referential word removal, addition of named entities and pruning. The thesis also describes the changes and additions made to facilitate running the system on Arabic and Chinese in addition to English, as this was the task for the shared task of Pradhan et al. (2012).

.

Acknowledgements

I would like to thank Pierre Nugues for tremendous help in completing this, and for much appreciated aid, both during the work and the writing of this report.

I would further like to thank Peter Exner for much valuable insight into the field of NLP.

And all the people in "exjobbsrummet" should be thanked thoroughly too. Yes. Very much so indeed. So. Thanks. A lot. For, well, everything.

Contents

Chapter 1

Introduction

The process of co-reference resolving is one of trying to find words or phrases, **mentions**, that are referring to the same thing or entity, chain them together and form a collection of such mentions. A mention during the scope of this text will refer to a word or phrase that is part of such a chain, in the manually annotated corpora described in Section 1.3, and a **markable** will refer to a candidate that will be processed and possibly become a mention.

There are a variety of ways to attempt coreference resolving. The most obvious and by far the most accurate is using humans to read and annotate a text manually. This is, however very time consuming and thus expensive. Computer aided methods have been devised and are currently under development, of which many utilize machine learning.

In the example text in Table 1.1, a portion of the data from the year 2011 can be seen in a simplified version with some columns left out. In the example there are words and phrases that corefer, for instance the phrase *Vandenberg and Rayburn* that corefer with the word *They* (coref id 8). For a human reader it is obvious that the words represent the same object in the real world but to a computer the task is a non-trivial one.

As a human reader you would use real world knowledge on which to base an assumption that the words actually relate to each other but a computer does not have this knowledge and thus other techniques need to be applied. A first step is to generate more information about each word in the sentence, part of speech (POS) and various other syntactical and semantic information that can be extracted and created using pre-existing tools.

Indx	Word	POS	Parse bit	Named Entities	Coref.
0	“	“	(TOP(S(S*	*	-
1	Vandenberg	NNP	(NP*	(PERSON)	(8|(0)
2	and	CC	*	*	-
3	Rayburn	NNP	*)	(PERSON)	(23)|8)
4	are	VBP	(VP*	*	-
5	heroes	NNS	(NP(NP*)	*	-
6	of	IN	(PP*	*	-
7	mine	NN	(NP*))))	*	(15)
8	,	,	*	*	-
9	”	”	*)	*	-
10	Mr.	NNP	(NP*	*	(15
11	Boren	NNP	*)	(PERSON)	15)
12	says	VBZ	(VP*	*	-
13	,	,	*	*	-
14	referring	VBG	(S(VP*	*	-
15	as	RB	(ADVP*	*	-
16	well	RB	*)	*	-
17	to	IN	(PP*	*	-
18	Sam	NNP	(NP(NP*	(PERSON*	(23
19	Rayburn	NNP	*)	*)	-
20	,	,	*	*	-
21	the	DT	(NP(NP*	*	-
22	Democratic	JJ	*	(NORP)	-
23	House	NNP	*	(ORG)	-
24	speaker	NN	*)	*	-
25	who	WP	(SBAR(WHNP*)	*	-
26	cooperated	VBD	(S(VP*	*	-
27	with	IN	(PP*	*	-
28	President	NNP	(NP*	*	-
29	Eisenhower	NNP	*))))))))))	(PERSON)	23)
30	.	.	*))	*	-
0	“	“	(TOP(S*	*	-
1	They	PRP	(NP*)	*	(8)
2	allowed	VBD	(VP*	*	-
3	this	DT	(S(NP*	*	(6
4	country	NN	*)	*	6)
5	to	TO	(VP*	*	-
6	be	VB	(VP*	*	(16)
7	credible	JJ	(ADJP*)))))	*	-
8	.	.	*))	*	-

Table 1.1: An example from the English corpus used in the 2011 CoNLL evaluation Pradhan et al. (2011). This example will be used throughout the rest of the thesis. For simplicity and ease of reading, this example does not contain all the layers of annotation.

1.1 Background

In many applications it is interesting to link mentions to real world entities. For instance, linking a mention of a person to a wiki page related to the person, or being able to create a hyperlink to a website of a corporation.

There are a number of ways to be able to create such links. From very basic string matching techniques to more sophisticated ones utilizing machine learning. But, it's not always possible to find all the mentions in a text that should link to the same entity.

Using the example *The author of this document is Marcus Stamborg. He can be difficult to locate sometimes.* it would require additional processing to be able to link *He* to *Marcus Stamborg*, more precisely it would require coreference resolving.

1.2 CoNLL Evaluation

Every year since 1999, a competition (a shared task) has been held in conjunction with the EMNLP-CoNLL conference (EMNLP-CoNLL 2012: Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning) to allow participants the ability to compare their results in various fields pertaining to natural learning techniques in a fixed and comparable setting. In 2011, the shared task (Pradhan et al., 2011) described the problem of coreference resolving in English. In 2012, the task was extended to include Arabic and Chinese as well, which adds another level of complexity as the system should preferably be able to handle these other languages seamlessly. Due to the languages being very different, we tried to find language independent ways of finding the coreference chains.

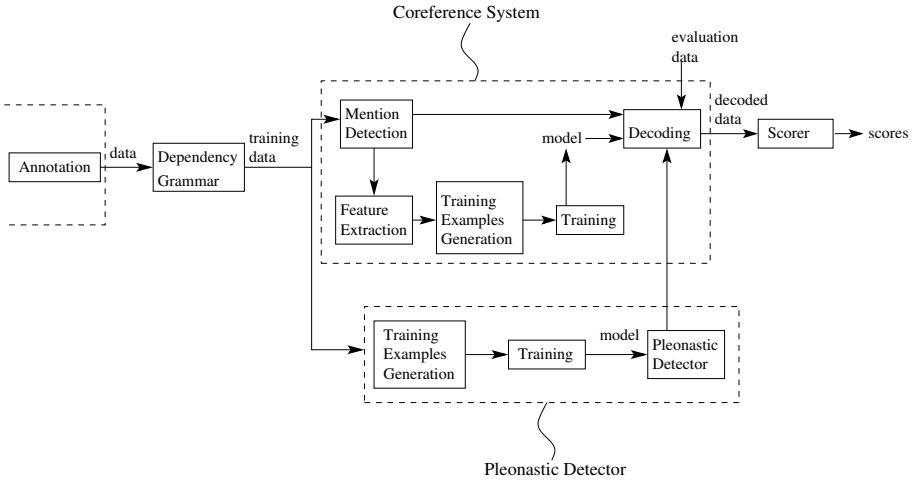

Figure 1.1: A general overview of the flow of the system. The annotation, the first step, was done by the organizers of the CoNLL evaluation (Pradhan et al., 2012). No changes to the flow was done except for the pleonastic detection in the bottom of the chart.

The input data for the system was given as part of the competition and

contains annotated documents in a specific format described in the next subsection. The input data is annotated by the organizers of the CoNLL evaluation (Pradhan et al., 2012) before it is given to the participants. The annotation step in Fig 1.1 denotes this process. This data is used by a machine learning algorithm (Fan et al., 2008) to train a model that is used to make the decision whether a mention corefer with another mention or not during the decoding stage of the process. A coreference chain is a collection of mentions that are coreferent according to some source, be it a human annotator or a computer.

One big problem regarding coreference resolving is how to compare different techniques. There is no universal consensus regarding the metric to use in measuring the accuracy of a system. It's only recently that there has been a large corpus containing coreference information, and the CoNLL evaluation is using a combination of three metrics to calculate the official score. These metrics and some background to them will be described further in Chapter 2.

1.3 CoNLL Format

The input data used in the competition contains material from the OntoNotes project (Hovy et al., 2006), specifically the portion of the OntoNotes data that is annotated with all the layers of information. An annotated file will henceforth be called a corpus.

The English corpus consists of over 1,500,000 words of which approximately 450,000 words are from newswire, 150,000 from magazine articles, 200,000 from broadcast news, 200,000 from broadcast conversations, roughly 200,000 words from web sources, 200,000 from telephone conversations and an additional 200,000 words from the English translation of the New Testament of the Bible.

The Chinese corpus contains a little over 1,000,000 words segmented as approximately 300,000 words from newswire, 300,000 from broadcast news, 200,000 from broadcast conversation, 200,000 from web sources and roughly 100,000 words from telephone conversations.

The Arabic corpus only contains around 300,000 words from newswire and as such it is a small and unbalanced source.

The words in the Chinese and Arabic corpora are written in their respective languages.

The data in the corpus is segmented in layers denoting various types of information as can be seen in Table 1.2.

In the example above, the word *Disney* and the word *it* are coreferent as can be discerned by looking at the last column. Each corpus is divided into two separate files, one for training and one for evaluation. The training part is 10 times larger and both files contain documents evenly selected from all the available sources to make sure the corpora are as unbiased as possible.

In addition to automatically tagged corpora, apart from the coreference information which is always hand-annotated, a gold-annotated version of each corpus was given. These gold-annotated corpora are manually annotated versions that should be error free, but mistakes can exist in these due to human error. It was permitted to use the gold versions as well as the automatically annotated versions but we've only used the automatically tagged versions, apart from extracting a dictionary for Arabic which is described further in section 4.3.

Column	Type	Description
1	Document ID	This is a variation on the document filename
2	Part number	Some files are divided into multiple parts numbered as 000, 001, 002, ... etc.
3	Word number	The first word in a sentence is 0
4	Word itself	This is the token as segmented/tokenized in the Treebank.
5	Part of Speech	For example, NN denotes a noun and VB denotes verb
6	Parse bit	The sentences are broken down into constituents and represented as a bracketed structure broken before the first open parenthesis in the parse, and the word/part-of-speech leaf replaced with a $*$. The full parse can be created by substituting the asterix with the "([pos] [word])" string (or leaf) and concatenating the items in the rows of that column.
7	Predicate lemma	The predicate lemma is mentioned for the rows for which we have semantic role information. All other rows are marked with a "-"
8	Predicate Frameset ID	This is the PropBank frameset ID of the predicate in Column 7.
9	Word sense	This is the word sense of the word in Column 3.
10	Speaker/Author	This is the speaker or author name where available. Mostly in Broadcast Conversation and Web Log data.
11	Named Entities	These columns identifies the spans representing various named entities.
12:N	Predicate Arguments	There is one column each of predicate argument structure information for the predicate mentioned in Column 7.
N	Coreference	Coreference chain information encoded in a parenthesis structure.

Table 1.2: The annotated layers and their description in the corpora. This information is taken directly from http://conll.cemantix.org/2012/data.html with some additional information by myself

1	2	3	4	5	6	7	8	9	10	11	12	13	14
bc/cctv/00/cctv_0000	0	0	The	DT	(TOP(S(NP(NP∗	-	-	-	Zhou_liangshuyi	∗	(ARG1∗	∗	-
bc/cctv/00/cctv_0000	0	1	most	RBS	(ADJP∗	-	-	-	Zhou_liangshuyi	∗	∗	∗	-
bc/cctv/00/cctv_0000	0	2	important	JJ	∗)	-	-	-	Zhou_liangshuyi	∗	∗	∗	-
bc/cctv/00/cctv_0000	0	3	thing	NN	∗)	thing	-	1	Zhou_liangshuyi	∗	∗	∗	-
bc/cctv/00/cctv_0000	0	4	about	IN	(PP∗	-	-	-	Zhou_liangshuyi	∗	∗	∗	-
bc/cctv/00/cctv_0000	0	5	Disney	NNP	(NP∗)))	-	-	-	Zhou_liangshuyi	(ORG)	∗)	∗	(50)
bc/cctv/00/cctv_0000	0	6	is	VBZ	(VP∗	be	01	2	Zhou_liangshuyi	∗	(V∗)	∗	-
bc/cctv/00/cctv_0000	0	7	that	IN	(SBAR∗	-	-	-	Zhou_liangshuyi	∗	(ARG2∗	∗	-
bc/cctv/00/cctv_0000	0	8	it	PRP	(S(NP∗)	-	-	-	Zhou_liangshuyi	∗	∗	(ARG1∗)	(50)
bc/cctv/00/cctv_0000	0	9	is	VBZ	(VP∗	be	01	1	Zhou_liangshuyi	∗	∗	(V∗)	-
bc/cctv/00/cctv_0000	0	10	a	DT	(NP∗	-	-	-	Zhou_liangshuyi	∗	∗	(ARG2∗	-
bc/cctv/00/cctv_0000	0	11	global	JJ	∗	-	-	-	Zhou_liangshuyi	∗	∗	∗	-
bc/cctv/00/cctv_0000	0	12	brand	NN	∗)))))	brand	-	1	Zhou_liangshuyi	∗	∗)	∗)	-
bc/cctv/00/cctv_0000	0	13	.	.	∗))	-	-	-	Zhou_liangshuyi	∗	∗	∗	-

Table 1.3: An example of the format given in the English corpus with all layers preserved. The numbers at the top corresponds to the layers in Table 1.2.

In order to be able to find coreferent mentions it is necessary to know what the mentions are in order to feed these to the system. A first step towards a coreference resolver is to extract a set of candidates known as markables. These are words and groups of words that could possibly be part of a coreference chain.

For the remainder of the report I will not use the full annotation when displaying examples. Most column will be left out in order to remove unnecessary information.

1.4 Aim of the thesis

The goal with this work is to present a number of improvements to the base system (Björkelund and Nugues, 2011) and our participation in the CoNLL-2012 shared task (Pradhan et al., 2012). This report will briefly describe the original system but the focus will be on the additions and improvements. When referring to additions I refer to the additions to the system made by pleonastic *it* removal, named entity addition and pruning of shorter mentions. These additions will all be described further in the report.

1.5 Related work

Much of the contemporary work in this field is collected in the CoNLL-2011 (Pradhan et al., 2011) and Conll-2012 (Pradhan et al., 2012) proceedings. During the 2012 EmNLP-CoNLL conference when the papers describing the participating systems were released it became clear that Anders Björkelund and his team had improved on the original system in similar ways but had also managed to increase the score even further and the system built by Björkelund and Farkas (2012) got the second place in the competition.

1.5.1 Soon's Pairwise Classification

There are a number of ways to perform coreference classification.

Soon et al. (2001)'s algorithm is based on pairwise classification. A set of features is extracted from pairs of mentions. This extracted information is used to train a model. This model is then used during decoding. During training it is necessary to have both positive and negative examples.

A positive example is generated by taking each anaphoric mention and pairing it to its closest preceding antecedent. This pair constitutes a positive example and a set of negative examples are created by pairing the **anaphor** with the mentions between the anaphor and the **antecedent**.

The anaphor is the word or phrase that refers to an antecedent. In the example *Today, Marcus is presenting his project.* the word *his* is the anaphor and *Marcus* is the antecedent.

The pairwise method can seem a bit naive as it only considers pairs as opposed to entire chains.

The LTH coreference classifier is based upon Soon et al. (2001)'s algorithm.

1.5.2 Rule-based Classification

For English especially, there are systems in the top tier that use a completely rule-based approach or a combination of rules and machine learning. In the CoNLL-2011 shared task, a completely rule-based system (Lee et al., 2011) won and was consequently regarded as state-of-the-art.

In a ruled-based system every aspect of the classification is based on grammatical and semantic rules. For instance, there could be a rule that determines whether two mentions share the same gender, number and if the strings representing the words match exactly. A rule-based classifier comprises various rules as simple as in the example, as well as ones of much greater complexity that requires a substantial amount of language specific knowledge.

Chapter 2

Metrics

As part of the competition we were given a scorer, a script that takes our systems output as input and calculates the recall, precision and F1-score. It also calculates five values, scores, based on five different scoring algorithms. When it comes to coreference, the researchers have not been able to come to a consensus regarding the metric to use for scoring a systems result. This makes it difficult to compare systems based on using one specific metric with another using a different metric.

The CoNLL 2012 evaluation (Pradhan et al., 2012), as well as previous instances of the competition, is a way to be able to more accurately compare different systems. Instead of using a multitude of different metrics, the evaluation is an attempt to decide the current state of the art in various fields regarding natural language processing.

All the metrics have pros and cons and using a combination is an attempt to create a metric that is stable and can be used as a fair comparison of the systems. The official evaluation score is a combination of three of the metrics, chosen to try to create a score that is broad and stable. The metrics included in the official score is shown in Table 2.1.

I will not describe the five metrics in detail, but I'll give a brief overview of the metrics used and how the systems were scored.

Metric	Description
MUC	— Message Understanding Conferences (Vilain et al., 1995) — Link based metric. Counts the number of links in common between the reference mentions and the system mentions. Flawed in that you could put all the mentions in the same chain and receive a very high score as this only counts as a recall error. Putting one mention in the wrong chain, however, counts both as a recall and a precision error.
B^3	— B-cube (Bagga and Baldwin, 1998) — Link based metric. This metric is an attempt to improve the MUC score. Calculates precision and recall for each mention and includes singleton chains as well. Including singletons introduce a flaw, since the singletons increase the scores considerably, making the results seem very high using this metric.
CEAF_M	— Constrained Entity–Alignment F–Measure (Luo, 2005) — This metric is an attempt to reduce the inflation of scores introduced by the B^3 measure, because the B^3 measure allows the mapping of a reference mention to more than one gold mention. In CEAF_M, a system mention can only be mapped to one reference mention, and the best match is used. The CEAF_M measure considers mentions and the links between them. This measure considers singleton mentions and produces somewhat inflated results.
CEAF_E	— Constrained Entity–Alignment F–Measure (Luo, 2005) — This is the same metric as the CEAF_M, only that it considers entire chains – entities – instead of mentions.
BLANC	— BiLateral Assessment of Noun-Phrase Coreference (Recasens and Hovy, 2011) — This is a variation of the Rand index. It is created to better fit the task of evaluating coreference by addressing many flaws found in the previous measurement algorithms. It too, has flaws and one main one is that in cases where there are many mention but very few that corefer. In this case the system will receive a very low score even though it might only miss a single coreference link.
$\frac{\text{MUC}+\text{B}^3+\text{CEAF\_E}}{3}$	The official score used to compare the systems.

Table 2.1: The metrics used in the evaluation.

Chapter 3

Precision and Recall

Two very important measures used throughout the document is the precision and recall. The metrics used by the scorer to evaluate the coreference system use these measurements and I will use them where applicable.

3.1 Precision

The precision denotes the likelihood an extracted item is relevant. For example, when extracting the markables, the precision indicates how relevant the extracted markables are, i.e. how likely they are to be mentions. The precision is calculated as

$$P = \frac{t_p}{t_p + f_p} \tag{3.1}$$

where t_p denotes the number of true positives and f_p denotes the number of false positives.

3.2 Recall

The recall describes the coverage of (in the instance of markable extraction) the extracted markables, that is, how many of the mentions are extracted as markables. It is calculated as

$$R = \frac{t_p}{t_p + f_n} \tag{3.2}$$

where t_p denotes the number of true positives and f_n denotes the number of false negatives.

3.3 F_1 score

The two measurements can be combined and this is called the F-score and the general form is

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R} \tag{3.3}$$

With $\beta = 1$ you get the harmonic mean of recall and precision and this is called the F_1 score

Chapter 4

Grammar

It is often interesting to look at the syntactic structure of sentences in order to extract information that can be used, either to find mentions or to be able to derive rules based on the structure. This chapter describes the *Dependency Grammar* step in Fig 1.1.

There are various ways of representing the syntactic structure of a sentence. One such way is to create a constituent tree and another is by creating a dependency grammar. A large amount of useful information can be extracted from the bracketed constituent structure given in the corpora (the parse bit column in Table 1.1).

The headword of a phrase is an important piece of information since many features utilize information based on this word. The head of a phrase, or a sentence, is the word that dictates the grammatical features of the phrase. I'll illustrate this by reusing part of the example text in Table 1.1.

In the phrase *Vandenberg and Rayburn are heroes of mine*, the word *are* is the head. The structure of the phrase is <participants> are <something>. The word *are* dictates what is possible to put on either side to form a grammatically correct sentence.

It would be cumbersome to create rules for extracting certain types of information from the constituent tree, since it would be necessary to create separate rules for each feature to find the headword. Because the corpora provided for the CoNLL evaluation (Pradhan et al., 2012) only contain a constituent grammar, all constituent trees are converted into a dependency grammar. This is done to provide additional information about the structure of the sentences, mostly because some information is easier to extract from the dependency grammar, for instance the headword. This conversion is done as a preprocessor step and is not a part of the coreference system. Instead it is necessary to manually run the tools used to create the dependency grammar. This will be further explained in Section 4.3.

4.1 Dependency Grammar

A dependency grammar, like a constituent tree, retains the word order but creates a tree where each word is a node with the root node being the entire sentence. Every word modifies, or are modified by another word or the root

node. This makes it very easy to find the head of a word simply by following the link upwards in the tree starting from a node.

The dependency grammar created from a part of the example in Table 1.1 is shown in Figure 4.1.

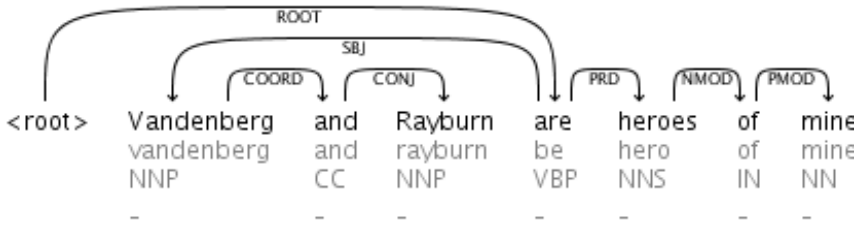

Figure 4.1: The dependency grammar with labelled edges denoting the type of modification performed by a word.

4.2 Constituent Trees

In linguistics, a constituent is one or more words that constitute a single group. That is, you should be able to substitute the group for another word and the sentence would still be grammatically correct.

The constituent tree retains the word order intact but makes it easy to identify various structures such as noun phrases and verb phrases. This makes it very easy to extract candidate mentions for instance. But if one would want to find the head of a word, a dependency grammar is more convenient.

The same segment of the example text above is displayed as a constituent tree in Figure 4.2.

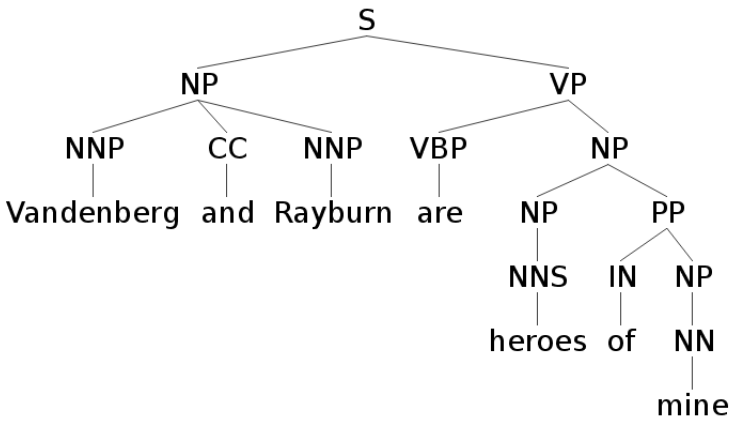

Figure 4.2: The constituent tree built from the parse bit layer of the annotated corpus.

Simplified tag set	Arabic tag set
NN	DET+NOUN+CASE_DEF_NOM
PRP$	NOUN+NSUFF_FEM_SG+CASE_INDEF_GEN

Table 4.1: The difference between the two versions of the tag set for Arabic.

4.3 Converting the Constituent Tree into a Dependency Grammar

Since it is easier to create some features and make them more versatile using a dependency grammar, every constituent tree is converted into a dependency grammar which is appended to the corpora. This is done, using automated tools, as a first step before feeding the corpus to the coreference system.

For English this was not a problem as tools were available to handle to conversion in the same way as in the original system. The LTH converter by Johansson and Nugues (2007) was used to convert the English corpus. For Chinese, the Penn2Malt converter by Nivre (2006) was applied using Chinese rules (http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html).

The Arabic conversion was a bit more problematic due to a discrepancy between the tag-set used in the Arabic corpus and the tag set expected by the CATiB converter by Habash and Roth (2009). The organizers of CoNLL Pradhan et al. (2012) had chosen to use a variation of the English tag set in the automatically tagged corpus instead of the full Arabic tag set.

The full Arabic tag set contains a wealth of information not found in the simpler English one. Information about gender, number, tense and more can be extracted from the rich Arabic tags. The difference can be seen in Table 4.1.

Fortunately, the gold version of the Arabic corpus was annotated using the full Arabic tag set which made it possible to extract a dictionary to convert the simplified version into the full Arabic version. This was done by means of cross validation, which is described in more detail in section 5.6.3 and creation of a dictionary.

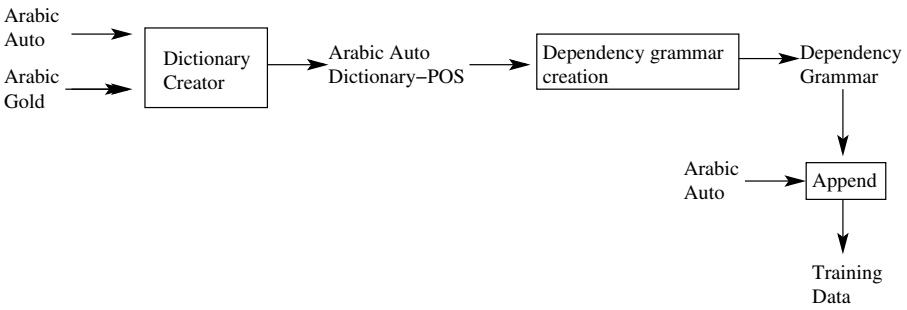

Figure 4.3: This is the flow of the conversion between constituent grammar and dependency grammar for Arabic. The grammar is created based on the gold version of the Arabic corpus but the grammar is appended to the automatically annotated version.

4.3.1 Dictionary

I created a dictionary by extracting the most common matching between the simplified and the full version and applying this information to the corpus before the conversion. In the final step of the conversion, one can choose which file to append the created dependency column to. By choosing the automatic version it is possible to receive a file with the original, simplified tag set but with a dependency graph created based on the full Arabic tag set. This flow is shown in Fig 4.3.

This dictionary is also used while running the system as some of the features were found to yield better results when given access to the full, more verbose tag set.

Chapter 5

System Architecture

5.1 Mention Detection

The corpus is processed one document at a time and the steps described are done for each document in turn. The first step that is performed is the extraction of markables. In the original system by Björkelund and Nugues (2011), this is done by extracting all noun-phrases (NP), pronouns (PRP) and possessive pronouns (PRP$). For English and Arabic the mention extraction is the same as in the original system, but for Chinese the addition of the PN tag is added. This tag denotes a pronoun in the Chinese tag set.

Going back to the example text in Table 1.1, I will show the mentions as extracted by the original system.

- Vandenberg and Rayburn
- heroes of mine
- heroes
- mine
- Mr. Boren
- Sam Rayburn , the Democratic House speaker who cooperated with President Eisenhower
- Sam Rayburn
- the Democratic House speaker who cooperated with President Eisenhower
- the Democratic House speaker
- President Eisenhower
- They
- this country

In the example, the mention detection will find that all the mentions are NP's even though the mention *They* could also be noted as a PRP. This is due to the extraction being recursive and it looks at the constituents before checking the terminals, that is, the single words themselves.

Further, the list of markables is not entirely correct. For instance, it is easy for a human to see that, for instance, *Vandenberg* should be a separate mention, as he is mentioned separately in the text. This is made further apparent by looking at the coreference ids in the last column. *Vandenberg* has a separate coreference id (0), while the phrase *Vandenberg and Rayburn* has the id (8).

I attempted to improve the mention detection by the utilization of named entity information and the pruning of shorter mentions. These improvements are described further in Chapter 8 and Chapter 9 respectively.

Other parts of the grammar could or perhaps should be included in the extraction since there are mentions that are not extracted by limiting the extraction to NP's, PRP's and PRP$'s. For instance, verb-phrases (VP) could participate in a coreference chain. Including VP's, however, is not as easy as simply adding the VP tag during extraction. The system would need to be adopted to handle verbs and to be able to pair them with other mentions. Due to time constraints and the fact that coreference chains including verb phrases are very rare in the documents, adaptations like this were not performed.

5.2 Linear Classification

In order to facilitate the computer making decisions whether a mention should be coreferent or not, it is necessary to train a model. The open source package LIBLINEAR (Fan et al., 2008) allows an easy way to do this. I chose to use this package both because of the fact that it was the package used in the original system and that it had proven to be stable and produce good results. Other means of training and decoding could be chosen, for instance the Weka library (Hall et al., 2009). The LIBLINEAR package (Fan et al., 2008) contains eight different algorithms to perform linear classification of which three perform logistic regression which are the only ones providing probability outputs.

Having a probability output as opposed to a Boolean output means that we can use the output to compare the classified pairs of mentions to each other which is a feature used extensively throughout the system.

In order to integrate the process of training and classifying, a Java port (Waldvogel, 2008) is used in the code. This enables training and classifying to be performed without human interaction and multiple runs with varying input can be run automatically. Since training and classifying takes time, it was beneficial to be able to make, sometimes almost a hundred, several runs of training and classifications during weekends, for instance to fine-tune a parameter.

When using logistic regression the probabilities of the possible outcomes are returned and one can choose the most probable one or opt for further analysis and use the probabilities in more intricate ways, as will be shown in Chapter 7.

Experiments, taking into account both time and the score when evaluating the coreference result, resulted in the selection of two algorithms from the LIBLINEAR package (Fan et al., 2008). The "L1-regularized logistic regression" algorithm during testing and the "L2-regularized logistic regression (dual)" algorithm during evaluation. The choices were made due to the first selection taking

less than a tenth of the time to create a model and the second one because it is more accurate for our data and thus yields better results. After experimenting with the two solvers I was confident that gaining an increase in the less accurate algorithm would also produce an increase using the more accurate one.

5.2.1 Features

A feature is something that gives information about an object, something that can be used to describe the object or its relation to another object. In the system, and the type of features that are compatible with LIBLINEAR (Fan et al., 2008), there exist three types of features and I will describe one feature for each type to give an idea of how features are constructed and used.

Boolean ex. StringMatch

Nominal ex. AnaphorPOS

Numeric ex. LongestCommonSubsequence (LCS)

A Boolean feature can have two values, true or false, where a nominal value can take a finite number of distinct values and the numeric can take any real number.

In order to use these features with LIBLINEAR (Fan et al., 2008) they need to be written in vector form where each possible value corresponds to a specific index in the feature vector. Given this, it is easy to imagine that vectors can become very large. Take, for instance, a feature simply holding the string value of a word, i.e. the word itself. In order to encode this as a vector, it is necessary to first extract all the possible words from the text and retain the word order from training to decoding in order to properly mask the correct index in the vector, the index corresponding to the specific word. For each feature only a single position in the vector is true making the vectors very sparse.

According to the documentation for the LIBLINEAR package (Fan et al., 2008), it performs best if the numeric values are normalized to a value between 0 and 1. This is done in some of the features used in the system but not all due to the score being lowered when applying normalization in some cases.

Boolean Feature

A Boolean feature is, as the name implies, a feature indicating a true or false relation and it creates a feature vector of size one.

The StringMatch feature is an example of a Boolean feature used in the system. It compares two strings and checks if they are exactly equal or not.

Nominal Feature

A small example of a nominal feature, more suitable in size for representation on paper, is using the POS tag from the example text in Table 1.1. I will use a subset of the possible POS tags found in the text and present one way of representing these as a feature vector.

In the text there are sixteen different tags but I'll limit the selection to four in order to make the feature vector easier to represent on paper.

Mentions	NNP	NN	PRP	VBD
(Eisenhower, mine)	0	1	0	0
(mine, They)	0	0	1	0

Table 5.1: Example vectors of the nominal feature AnaphorPOS. The first word is the antecedent and the second word is the anaphor.

A nominal feature using the POS tag is the AnaphorPOS feature. It extracts the POS tag of the anaphor. As seen in Table 5.1, every possible POS tag is represented by a specific position in the vector.

Numeric Feature

A numeric feature creates a vector of size one containing a real value. One numeric feature used in the system is the *Longest Common Subsequence* (LCS) which calculates the longest consecutive amount of characters in one string that is also present in another. The feature produces a real number which is stored in the vector for use during training. The result can be used as is or be normalized. In the case of the LCS feature it is normalized by dividing the result by the length of the two strings combined as,

$$\text{Normalized result} = \frac{\text{LCS}}{|string_1| + |string_2|}$$

5.3 Training examples

In order for the computer to learn which class a certain object should be classified as, it needs training examples. There need to be both positive and negative training examples in order for the training algorithm to be able to learn how to classify instances. It also allows for the the creation of a **model**. The model is a representation that is stored as a file and is used in the decoding step. The training is done on the training examples in a way that it will be possible to make a decision, even if you try to classify a previously unseen combination of values.

The training examples are extracted in the same manner as described by Soon et al. (2001). For each positive example consisting of adjacent and coreferent mentions Pos = $\{(M_i, M_k)\}$, we construct the negative examples by pairing M_k with each mention between M_i and M_k as Neg = $\{(M_j, M_k)|i < j < k\}$.

As positive pairs we are only interested in adjacent mentions that corefer. This means that if having three mentions that corefer, only two positive examples would be created. Having the mentions a_1, b_2, c_1, d_2, e_3 and f_1 the positive pairs would be $[a_1, c_1]$, $[c_1, f_1]$ and $[b_2, d_2]$. The subscript indicates the coreference id.

For each positive pair, the anaphor is paired with each mention between the anaphor and antecedent. The negative examples for each positive pair is shown

- Positive $[a_1, c_1]$
  - Negative $[b_2, c_1]$

- Positive [c_1, f_1]
  - Negative [d_2, f_1]
  - Negative [e_3, f_1]
- Positive [b_2, d_2]
  - Negative [c_1, d_2]

Creating the training examples in this fashion produces a significantly larger amount of negative training examples. The number of positive and negative examples per language is presented in Table 5.2

Language	Positive	Negative
English	109,098 (7.8 %)	1,294,689 (92.2 %)
Chinese	53,088 (5 %)	1,011,404 (95 %)
Arabic	15,816 (3.2 %)	474,872 (96.8) %

Table 5.2: The amount of positive and negative training examples per language as extracted from the training sets.

5.4 Feature Extraction

During the feature extraction phase, the previously extracted list of training examples is processed. A set of features is applied on each training example and necessary information is extracted during the process. For instance, a list of every possible lexical value, i.e. the words, is stored in order to be able to map the words to a specific position in the feature vector.

After the feature set has been applied to all the training examples, the set of features is saved to a file. This is done in order to be able to use the exact same set during both training and decoding.

5.5 Training

The training is performed by the LIBLINEAR package (Fan et al., 2008) by calling the appropriate methods in the library. The training results in a model file that can be used later to automatically decide the probability that a pair of mentions are coreferring.

5.6 Decoding

In the decoding step, we use the previously created model file along with the same combination of features as during training, but without the class indicator. The class indicator is what tells the system whether two mentions corefer during training, but during decoding it is not present as it is unknown.

The output we receive from the algorithm is a vector of probabilities. The vector will have two values indicating the probability that two mentions corefer, where the first value in the vector indicates the probability that the mentions corefer. The second value is the probability that they do not corefer. More precisely the vector holds the values $[P_{coref}(Antecedent, Anaphor), 1 - P_{coref}(Antecedent, Anaphor)]$.

The decoding phase is where the system calculates the probability that the markables in a pair are coreferent. This is done in two separate ways depending on the language. In the original system, more than one algorithm for decoding was implemented. Initially the same method was applied across the languages. After changing and improving the system substantially I ran the system for each language using different decoding algorithms and found that the original algorithm was not the optimal one to use for English.

5.6.1 Chinese and Arabic

Depending on the languages, we applied different decoding strategies: For Chinese and Arabic, we used a closest-first clustering method as described by Soon et al. (2001) for pronominal anaphors (he, she, it etc.) and a best-first clustering otherwise as in Ng and Cardie (2002).

The closest-first method starts with an anaphor and calculates the probability of coreference for each antecedent until it finds one antecedent where the probability output from the classifier is higher than the probability cut-off of 0.5. This selected antecedent is then added to the chain containing the anaphor.

The best-first method calculates the probability for all anaphor–antecedent pairs and select the antecedent with the highest probability of being coreferent.

5.6.2 English

For English, we applied a closest-first clustering for pronominal anaphors. For non-pronominal anaphors, we used an averaged best-first clustering. The averaged version does not only look at separate mentions but instead the already created chains. Given an anaphor, all chains before the anaphor are considered. The average probability of the anaphor being coreferent with each mention in the chain is calculated. This average value for each chain is then compared and the anaphor is added to the highest scoring chain if there exist a chain with an average score greater than 0.5. If no such chain is found then the algorithm performs a normal best-first clustering.

5.6.3 Cross-validation

In order to minimize the risk of overfitting the model to the evaluation data, 5-fold cross-validation was applied. This means that the training data was split into 5 equal parts with each part containing roughly the same amount of data from each source, newswire, broadcast etc. After this the system is trained 5 times to create the models. Each model is trained from four of the parts. The part that was left out from each model is then used as evaluation data for the model where it was left out. The scores from each run are used to calculate the arithmetic mean. This averaged score is used to compare the score when testing the additions and changes to the code as well as to optimise settings.

The reasoning behind using cross-validation is to lower the probability that the system would perform well on a specific batch of test data but poorly on others.

Chapter 6

Global Multi-language Adaptations

6.1 Number and Gender

As a part of the resources provided by the organizers of CoNLL-2012 (Pradhan et al., 2012), a lookup table (Bergsma and Lin, 2006) for number and gender was provided. By using this table it is possible to find whether a word is singular, plural or non-countable; feminine, masculine or neutral. This compilation was only given for English however, thus the part of the code that handles this lookup had to be divided into three separate parts, each taking care of one individual language.

The code pertaining to the English number and gender lookup was left mostly untouched, apart from minor bug-fixes and improvements to the code. For Chinese and English we made substantial changes. Instead of using a pre-compiled lookup table, myself with the aid of Peter Exner translated Chinese and Arabic pronouns, titles and various other words. These words were added to a list that would later be used instead of a pre-compiled table. Unfortunately this work only had a minor impact on the system score, most likely due to the features using this information only having a minor impact on the score themselves. It could also be that the translations are not sufficient and only enables the capture of a small amount of information.

6.2 Configuration files

All additions, pleonastic detection, Named Entity addition and pruning can be turned on or off by editing a configuration file. One such configuration file is created per language and more can be created to have permanent configurations with varying settings.

Chapter 7

Pleonastic it

7.1 Description

In a sentence there can exist words that are normally used to refer to another mention. The word *it* is one such word that is sometimes used as a non-referential *dummy*-word. In the example *Today, Marcus is presenting his project. It is snowing.* the word *It* does not refer to anything. It is used used as a pleonastic word. It is not very difficult for a human reader to see that the word *It* in the example above does not refer to another thing, but for a computer it is not trivial as this word contains many of the characteristics of an ordinary pronoun.

If the example was *Today, Marcus is presenting his project. It is about coreference resolving.* The word *It* would refer to *his report*. The distinction between the two versions is not easy for a computer to discern. Words with this characteristics is detrimental to the performance of the coreference system, as they can be used to link mentions that have nothing else in common. In doing so two chains that would be separate without the spurious *it* are merged and instead of having two correct chains you get one faulty chain.

In this chapter I will describe the work I've done in order to attempt to classify the word *it* as either pleonastic or not as it occurs in the English corpus.

7.2 Previous Work

The idea to remove pleonastic *it*s has been used in a couple of coreference solvers. An example is the high-performance Stanford solver (Lee et al., 2011) that includes a rule-based module to identify these pronouns. The rules consider the current word and the word following in the sentence. If the current word is *it* and any of the following words:

> *is*, *was*, *seems*, *seemed*, *appears*, *looks*, *means*, *follows*, *turns*, *turned*, *become*, *became*,

is found immediately after it, it is tagged as pleonastic and discarded from the mention list.

7.3 Classifier

To design a classifier, I used the approximation that non-coreferring pronouns, i.e. pronouns not member of a coreference chain in the annotated corpus were pleonastic. By definition, pleonastic pronouns are outside coreference chains. However, this idea failed to identify singleton pronouns that lack antecedents.

I trained a classifier using logistic regression and the LIBLINEAR package (Fan et al., 2008). For the training I created a small number of features on which a simple greedy forward/backward selection algorithm was run. Two features, **HeadLex** and **NextWordLex** were selected at this stage in the development. From the complete original coreference solver I could observe a slight increase of the score.

7.3.1 Pre/Post-processing

In the initial trials, a preprocessor was used to remove the pleonastic pronouns from the mentions. I also tried to move the removal to a post-processing stage, where the pronouns were discarded from the coreference chains. Although giving an increase of the overall score, it was lower than by using the preprocessor and I did not follow this path. The flows for both versions are shown in Fig 7.1.

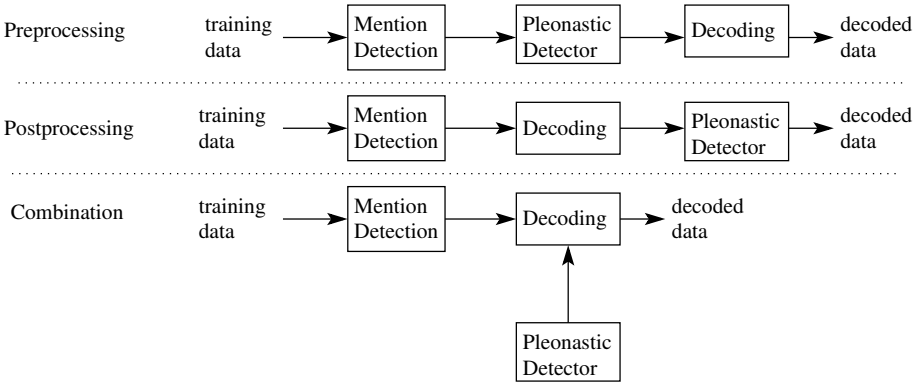

Figure 7.1: This is the flow of the three types of pleonastic *it* removal versions that were attempted.

7.3.2 Combination of Probabilities

I noticed that isolated pleonastic identifier modules, either as pre or post-processors, removed a significant portion of non-pleonastic *it*s. I introduced a second term in the classifier to take into account the likelihood that the pronoun was part of a coreferring pair. The flow for this is illustrated in Fig 7.1. I used the probability that the word was pleonastic, P_{pleo}, together with the result from the coreference resolver, P_{coref}. The inequality:

$$P_{coref}(\text{Antecedent}, it) \cdot (1 - P_{pleo}(it)) > 0.4,$$

was applied to decide on the pleonastic nature of *it*. The cut-off value of 0.4 was found experimentally, using 5-fold cross-validation.

English 2011	CoNLL score
Baseline	53.27
Rules	53.21
Pre-processor	53.51
Post-processor	53.63
Combination	53.90

Table 7.1: Scores on the 2011 English development set (Pradhan et al., 2011) using various ways of removing pleonastic *it* pronouns.

English 2012	CoNLL score
Without removal	59.15
With removal	**59.57**

Table 7.2: Scores on the 2012 English development set (Pradhan et al., 2012) with and without removal of the pleonastic *it* pronouns.

Using the probability combination, I carried out a second feature selection and three features was selected: **HeadLex**, **HeadRightSiblingPOS** and **HeadPOS**.

7.3.3 Pleonastic *it* Results

We carried out the initial testing with the 2011 CoNLL shared task corpus (Pradhan et al., 2011) and the original coreference system by (Björkelund and Nugues, 2011). Table 7.1 shows the results for the various alternatives we tested.

The system we submitted to CoNLL 2012 is significantly different, notably because it handles multiple languages and it uses a different feature set for the coreferring pair identification. Table 7.2 shows the final scores we obtained with this system on the development set with and without the pleonastic identification. We report the results on the CoNLL 2012 corpus which is slightly different

Status of *it*	Count	
Coref, Dev-set:	792	
Not Coref, Dev-set:	610	
Total *it*:	1402	
Set	With rem.	W/o rem.
Coref, Out-file:	966	1318
Not Coref, Out-file:	436	84
False Positives:	327	556
False Negatives:	153	30

Table 7.3: Statistics for the development set with and without pleonastic *it* removal. The entries detailing Dev-set are the number of referent and non-referent *its* in the input file. The entries for the out-file detail the number of *its* being classified as either coreferent or not by the system.

from the CoNLL 2011 corpus.

As can be seen in Table 7.3, the number of false negatives increase while the number of false positives decrease. Since the module only uses three features it is probable that it will remove instances that shouldn't be removed. But as these pleonastic *its* are detrimental to the coreference system it seems better to remove too many than too few.

Chapter 8

Named Entities

A named entity is a mention that refers to a specific, named, entity. For instance, Lund University, Marcus Stamborg or Sweden. In the corpus, named entities are tagged with a label to indicate what type of entity it is, for instance: ORGANIZATION, PERSON, LOCATION etc. In the initial corpora received from the organizers of the CoNLL evaluation, this information was available for all three languages. This was due to the organizers only making available the gold annotated versions of the Chinese and Arabic corpora initially. Once we received the automatically annotated versions, the information regarding named entities was was missing from the Chinese and Arabic corpora.

This means that the following description is only applicable on the English corpus and for results obtained by running the system on it.

Since the original system did not utilize the information found by looking at the named entity information I attempted to improve the recall of the mention detection by not only extracting mentions based on word classes, but also based on the named entity information.

The reason why the named entity information could be utilized is best explained by an example. In Table 8.1 the mention *Vandenberg and Rayburn* is extracted as a noun phrase. A human reader can see that the name *Vandenberg* and the word *Rayburn* could be referred to in the text and should thus be extracted as a mention, as shown in Table 8.2.

By utilizing the information in the named entity column of the data it is clear that both *Vandenberg* and *Rayburn* are tagged as persons.

A named entity can span several rows, including an entire phrase. In the first trial I limited the selection to only include named entities spanning a single row. By adding all of these single-row named entities, I saw I slight decrease in system performance. Due to this I decided to impose a rule to only include named entities spanning a single row, that were also inside a noun phrase. The example in Table 8.2 illustrates this.

Adding this rule had a major impact and I could see a very slight increase of the system score instead of a decrease.

Having found that using the named entity information to improve the mention extraction to be useful, I modified the extraction to allow the inclusion of mentions spanning more than one row. Allowing the machine learning algorithm to decide which information is useful proved to be the best course of action and I could see an increase of the system score by over 0.7 points.

Indx	Word	POS	Parse bit	Named Entities	Coref.
0	“	“	(TOP(S(S*	*	-
1	**Vandenberg**	NNP	**(NP***	(PERSON)	(8|(0)
2	**and**	CC	*****	*	-
3	**Rayburn**	NNP	***)**	(PERSON)	(23)|8)
4	are	VBP	(VP*	*	-
5	heroes	NNS	(NP(NP*)	*	-
6	of	IN	(PP*	*	-
7	mine	NN	(NP*))))	*	(15)
...					

Table 8.1:

Indx	Word	POS	Parse bit	Named Entities	Coref.
0	“	“	(TOP(S(S*	*	-
1	**Vandenberg**	NNP	(NP*	**(PERSON)**	(8|(0)
2	and	CC	*	*	-
3	**Rayburn**	NNP	*)	**(PERSON)**	(23)|8)
4	are	VBP	(VP*	*	-
5	heroes	NNS	(NP(NP*)	*	-
6	of	IN	(PP*	*	-
7	mine	NN	(NP*))))	*	(15)
...					

Table 8.2:

By removing specific named entity categories, that are guaranteed never to be part of a chain, the score improved further. The categories that are excluded are: Cardinal, Ordinal, Percent and Quantity.

The results from the addition of the named entity information can be seen in Table 8.3, which shows the official CoNLL score as well as the score without adding NE's.

English 2012	CoNLL score
With NE's	**59.57**
Without NE's	58.85

Table 8.3: Scores on the 2012 English development set (Pradhan et al., 2012) with and without the addition of named entities during the mention extraction phase.

Chapter 9

Pruning of Shorter Mentions

In the corpora there can be nested structures where, for instance, a noun phrase is contained within a larger noun phrase.

In the sentence *Marcus Stamborg, the author of the report* there are five mentions:

- Marcus Stamborg
- the author
- the report
- the author of the report
- Marcus Stamborg, the author of the report

Convention says that if two mentions share the same head, according to the dependency analysis, the shorter of the two should be discarded and consequently not considered as a mention. Out of the five extracted mentions, two would be discarded. *Marcus Stamborg* and *Marcus Stamborg, the author of the report* share the same head as does *the author* and *the author of the report*.

This results in the final list of mentions being:

- the report
- the author of the report
- Marcus Stamborg, the author of the report

Instead of rewriting the code that extract markables to look for this type of nesting I opted to create a standalone addition that could be turned on or off. Being standalone also makes it easy to modify and extend, should the need arise.

The impact of this addition varies across the languages. For English the impact is on par with the pleonastic *it* removal but for Chinese the impact is much greater, improving the system score by over 5.5 points. Arabic stand out due to the pruning lowering the system score.

The change to the scores on the final result when turning on or off the pruning can be seen in Table 9.1.

Setting	English	Chinese	Arabic
With pruning	**59.54**	**56.29**	47.10
Without pruning	56.42	50.94	**48.05**

Table 9.1: The official CoNLL scores for all languages and the result when turning off the pruning of shorter mentions.

9.1 Arabic

The conversion from the constituent grammar to the dependency grammar for Arabic is based on a dictionary. This most likely introduces errors in the dependency tree that might explain some of the poor performance when applying pruning on Arabic. The conversion tool itself is also an error source.

Another plausible explanation could be that the Arabic corpus was not annotated in this way, simply, that the convention was not followed. Thus by only keeping the longer markable the system could not perform as well due to not having access to the necessary mentions.

Since the organizers seemed to have some difficulty finding automatic tools to annotate the Arabic corpus, it is also possible that the annotation itself is flawed.

Another explanation could be the Arabic language having a very rich morphology, the order in which words appear in a sentence is more or less arbitrary.

Due to lack of time I decided to not press further into trying to find a solution and chose to run the system without pruning for Arabic.

Chapter 10

Results

This chapter lists the results from running the system using various settings.

10.1 Mention Extraction

When comparing the scores from runs with different settings regarding the markable extraction phase it is apparent that changing the recall and precision of the extracted markables has a great impact on the system performance. It is very important to have a high recall as this increases the score significantly, much more so than increasing the precision.

When applying pruning the recall lowers slightly compared to the recall when adding the named entities, however, the increase in precision is significant enough to more than compensate for this.

English

Setting	Precision	Recall	CoNLL Score
No additions	32.82	92.17	57.22
Named entities	31.61	94.47	56.42
Prune	39.55	90.52	58.43
NE's + prune	38.72	92.63	**59.15**

Table 10.1: The impact of recall and precision of the extracted mention during the mention extraction phase. All scores are with pleonastic *it* removal turned off.

Chinese

Setting	Precision	Recall	CoNLL Score
No additions	32.29	87.32	**50.94**
Prune	40.16	86.39	56.62

Table 10.2: The impact of recall and precision of the extracted mentions during the mention extraction phase. There is no named entity information for Chinese.

Arabic

Setting	Precision	Recall	CoNLL Score
No additions	17.64	87.23	**48.25**
Prune	20.42	78.57	47.10

Table 10.3: The impact of recall and precision of the extracted mentions during the mention extraction phase. There is no named entity information for Arabic.

10.2 Non-referential *it*

English 2011	CoNLL score
Baseline	53.27
Rules	53.21
Pre-processor	53.51
Post-processor	53.63
Combination	**53.90**

Table 10.4: Scores on the 2011 English development set (Pradhan et al., 2011) using various ways of removing pleonastic *it* pronouns.

English 2012	CoNLL score
Without removal	59.15
With removal	**59.57**

Table 10.5: Scores on the 2012 English development set (Pradhan et al., 2012) with and without removal of the pleonastic *it* pronouns. This table shows the scores with both pruning and named entity additions turned on.

English 2012	CoNLL score
Without additions	57.22
With *it* removal	**57.59**

Table 10.6: Scores on the 2012 English development set (Pradhan et al., 2012) with and without removal of the pleonastic *it* pronouns. This table shows the scores with both pruning and named entity additions turned off.

10.3 Named Entities

English

English 2012	CoNLL score
With NE's	**59.57**
Without NE's	58.85

Table 10.7: Scores on the 2012 English development set with and without the addition of named entities during the mention extraction phase. All other additions are turned on.

English 2012	CoNLL score
With NE's	56.42
Without NE's	**57.22**

Table 10.8: Scores on the 2012 English development set with and without the addition of named entities during the mention extraction phase. All other additions are turned off.

10.4 Pruning of Shorter Mentions

English

Setting	CoNLL Score
With pruning	**58.44**
Without pruning	57.22

Table 10.9: The official CoNLL scores for English and the result when turning off the pruning of shorter mentions. All other additions are turned off.

Chinese

Setting	CoNLL Score
With pruning	**56.62**
Without pruning	50.94

Table 10.10: The official CoNLL scores for Chinese and the result when turning off the pruning of shorter mentions.

Arabic

Setting	CoNLL Score
With pruning	47.10
Without pruning	**48.25**

Table 10.11: The official CoNLL scores for Arabic and the result when turning off the pruning of shorter mentions.

10.5 Coreference

Metric/Corpus	Development set			Test set		
English	R	P	F1	R	P	F_1
Mention detection	74.21	72.81	73.5	75.51	72.39	73.92
MUC	65.27	64.25	64.76	66.26	63.98	65.10
B^3	69.1	70.94	70.01	69.09	69.54	69.31
CEAF_M	57.56	57.56	57.56	56.76	56.76	56.76
CEAF_E	43.44	44.47	43.95	42.53	44.89	43.68
BLANC	75.36	77.41	76.34	74.03	77.28	75.52
CoNLL score	59.27	59.89	**59.57**	59.29	59.47	**59.36**
Chinese	R	P	F1	R	P	F_1
Mention detection	60.55	68.73	64.38	63.07	75.21	68.61
MUC	54.63	60.96	57.62	58.32	67.95	62.76
B^3	66.91	74.4	70.46	67.47	78.54	72.58
CEAF_M	55.09	55.09	55.09	58.98	58.98	58.98
CEAF_E	44.65	39.25	41.78	49.80	41.15	45.06
BLANC	73.23	72.95	73.09	76.29	80.09	78.05
CoNLL score	54.4	58.2	**56.62**	58.53	62.55	**60.13**
Arabic	R	P	F1	R	P	F_1
Mention detection	55.54	61.7	58.46	56.1	63.28	59.47
MUC	39.18	43.76	41.34	39.11	43.49	41.18
B^3	59.16	67.94	63.25	61.57	67.95	64.61
CEAF_M	47.8	47.8	47.8	50.16	50.16	50.16
CEAF_E	42.57	38.01	40.16	44.86	40.36	42.49
BLANC	62.44	67.18	64.36	66.80	66.94	66.87
CoNLL score	46.97	49.9	**48.25**	48.51	50.6	**49.43**

Table 10.12: Scores on the development set and the test set for English, Chinese, and Arabic. R is the recall, P the precision and F_1 is the harmonic mean. The official CoNLL score is computed as the arithmetic mean of MUC, BCUB, and CEAFE.

Chapter 11

Discussion

11.1 Discussion

The current system was an attempt to improve upon the existing coreference solver. It adds features not existing in the original version and parts of the code were rewritten to be able to handle languages other than English.

I first attempted to improve the score by looking at the word *it* and creating a classifier to detect when it is unlikely that one specific instance of the word is part of a coreference chain. This proved beneficial and the score was improved. I have not attempted to find words with similar characteristics for the other languages due to a a lack of time.

Next I looked at utilizing the named entity column in the data. This helped increase the recall during the markable extraction at the expense of the precision. It turned out that increasing the precision was more important and the addition of markables based on the named entity information helped to increase the score further.

I also added a module to look at the extracted mentions and removing any mentions nested inside larger, encompassing mentions. This is a convention, and the corpora are annotated according to this principle. By not pruning, the system considered mentions that, by convention, never should be considered. By performing the pruning the system produced chains that were cleaner, which improved upon the score for English and Chinese. Why the pruning lowers the score for Arabic is still a mystery, it could be that the annotation is different. In Arabic, the word order is somewhat arbitrary and this might influence the way nesting is done.

Arabic is also different, but in a way similar to Chinese, in that gender and number information is found by inflections of the words themselves, so in order to extract such information it would be necessary to have knowledge of the language.

Looking at the scores for the named entity addition it can seem a bit strange that the score is lowered when applying the named entity addition alone when it, in combination with pruning increases the score further. A system like this, with a multitude of features that are found using automatic feature searching, creates dependencies between all the parts of the system. During development, when I was trying out named entities, the score increased by adding mentions

based on that information. However, during the course of further development the system has adapted the features and at the current state it is necessary to have the additions turned on together to get the correct score. It would not be advisable to alter one part without performing another feature search after.

11.2 Conclusions

Given the average score across the three languages resulting in a fourth place in the CoNLL evaluation (Pradhan et al., 2012), it is shown that the system is adaptable to be able to cope with various languages. The detection of the non-referential *its* proved to be a valuable addition for English and extending it to other words would likely prove beneficial. I think that it would be necessary to redo parts of the application to be able to gain much higher scores, using methods proven to be valuable in other, contemporary works. Focusing mainly on feature engineering is difficult and the evaluation of features is computationally expensive as well as highly time-consuming.

It is difficult coming to any extensive conclusions regarding Arabic as the amount of information given in the corpus was low. All systems participating in the evaluation had a difficult time getting good scores.

11.3 Future Work

Extending the non-referential detector to words other than *it* for English and attempt to run a similar analysis on other languages to try to find words that might have the same properties would likely prove beneficial, at least for English.

I would have wanted to attempt to build a cluster model where the chains are built not by adding mentions separately. Instead, also making sure that the chains are sound in regard to not linking mentions to a chain where there are conflicts. For instance there could be a chain linking three mentions. A, B and C. Mentions A and B are coreferent and so is B and C. However, A and C are not coreferent, but they are placed in the same chain because of the other two links. I attempted to check the chains after they were created and finding mentions that would be unlikely to be part of the chain Even though I did manage to find such chains I could not get a positive result on the score. It would be interesting to see what results could be obtained by doing a more in-depth study. More precisely, to re-implement the decoding to perform this type of checks during, instead of after decoding.

An idea that formed during development that was never pursued, again due to time constraints, was to use different types of machine learning algorithms and try to see if they perform differently on different types of text. For instance one algorithm might perform best on newswire but poorly on other types. Another might perform well on broadcast types but poorly on newswire. The idea then, was to train models using different algorithms and apply them to the sections where they perform best.

Bibliography

Bagga, A. and Baldwin, B. (1998). Algorithms for scoring coreference chains. In *Proceedings of the LREC 1998 Workshop on Linguistic Coreference*, pages 563–566.

Bergsma, S. and Lin, D. (2006). Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Sydney, Australia. Association for Computational Linguistics.

Björkelund, A. and Farkas, R. (2012). Data-driven multilingual coreference resolution using resolver stacking. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 49–55, Jeju Island, Korea. Association for Computational Linguistics.

Björkelund, A. and Nugues, P. (2011). Exploring lexicalized features for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 45–50, Portland, Oregon, USA. Association for Computational Linguistics.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Habash, N. and Roth, R. (2009). CATiB: The Columbia Arabic treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations*, 11(1).

Hovy, E. H., Marcus, M. P., Palmer, M., Ramshaw, L. A., and Weischedel, R. M. (2006). Ontonotes: The 90 In Moore, R. C., Bilmes, J. A., Chu-Carroll, J., and Sanderson, M., editors, *HLT-NAACL*. The Association for Computational Linguistics.

Johansson, R. and Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In Nivre, J., Kaalep, H.-J., Muischnek, K., and Koit, M., editors, *NODALIDA 2007 Conference Proceedings*, pages 105–112, Tartu.

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M., and Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34, Portland, Oregon, USA. Association for Computational Linguistics.

Luo, X. (2005). On coreference resolution performance metrics. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 25–32.

Ng, V. and Cardie, C. (2002). Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Nivre, J. (2006). *Inductive Dependency Parsing.* Springer, Dordrecht, The Netherlands.

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., and Zhang, Y. (2012). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning (CoNLL 2012)*, Jeju, Korea.

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R., and Xue, N. (2011). CoNLL-2011 shared task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon, USA. Association for Computational Linguistics.

Recasens, M. and Hovy, E. (2011). Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17.

Soon, W. M., Ng, H. T., and Lim, D. C. Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D., and Hirschman, L. (1995). A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, MUC6 '95, pages 45–52.

Waldvogel, B. (2008). http://liblinear.bwaldvogel.de/.