KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second Cycle, 30 Credits

Coreference Resolution for Swedish

LISA VÄLLFORS

Stockholm, Sweden 2022

Coreference Resolution for Swedish

LISA VÄLLFORS

Degree Programme in Computer Science and Engineering
Date: March 8, 2022

Supervisor: Johan Boye
Examiner: Viggo Kann
School of Electrical Engineering and Computer Science
Swedish title: Koreferenslösning för svenska

Abstract

This report explores possible avenues for developing coreference resolution methods for Swedish. Coreference resolution is an important topic within natural language processing, as it is used as a preprocessing step in various information extraction tasks. The topic has been studied extensively for English, but much less so for smaller languages such as Swedish. In this report we adapt two coreference resolution algorithms that were originally used for English, for use on Swedish texts. One algorithm is entirely rule-based, while the other uses machine learning. We have also annotated a Swedish dataset to be used for training and evaluation.

Both algorithms showed promising results and as none clearly outperformed the other we can conclude that both would be good candidates for further development. For the rule-based algorithm more advanced rules, especially ones that could incorporate some semantic knowledge, was identified as the most important avenue of improvement. For the machine learning algorithm more training data would likely be the most beneficial. For both algorithms improved detection of mention spans would also help, as this was identified as one of the most error-prone components.

Keywords

Natural language processing, Information extraction, Machine learning, Random forests, Coreference resolution

Sammanfattning

I denna rapport undersöks möjliga metoder för koreferenslösning för svenska. Koreferenslösning är en viktig uppgift inom språkteknologi, eftersom det utgör ett första steg i många typer av informationsextraktion. Uppgiften har studerats utförligt för flera större språk, framförallt engelska, men är ännu relativt outforskad för svenska och andra mindre språk. I denna rapport har vi anpassat två algoritmer som ursprungligen utvecklades för engelska för användning på svensk text. Den ena algoritmen bygger på maskininlärning och den andra är helt regelbaserad. Vi har också annoterat delar av Talbankens korpus med koreferensrelationer, för att användas för träning och utvärdering av koreferenslösningsalgoritmer.

Båda algoritmerna visade lovande resultat, och ingen var tydligt bättre än den andra. Bägge vore därför lämpliga alternativ för vidareutveckling. För ML-algoritmen vore mer träningsdata den viktigaste punkten för förbättring, medan den regelbaserade algoritmen skulle kunna förbättras med mer komplexa regler, för att inkorporera exempelvis semantisk information i besluten. Ett annat viktigt utvecklingsområde är identifieringen av de fraser som utvärderas för möjlig koreferens, eftersom detta steg introducerade många fel i bägge algoritmerna.

Nyckelord

Språkteknologi, informationsextraktion, maskininlärning, beslutsträdsinlärning, koreferenslösning

Contents

List of Tables

List of acronyms and abbreviations

ACE Automatic Content Extraction (a research program)

CoNLL The SIGNLL Conference on Computational Natural Language Learning

ML Machine Learning

MUC The Message Understanding Conferences

NER Named Entity Recognition

NLP Natural Language Processing

PMI Pointwise Mutual Information

POS Part of Speech

SUC Stockholm-Umeå Corpus

Chapter 1

Introduction

Coreference resolution is the task of determining which entity mentions in a text refer to the same real-world entity. It is an important part of many tasks such as question answering, text summarization and machine translation. This is an example of coreference:

> $[\text{Alice}]_x$ gave $[[\text{her}]_x$ book$]_y$ to $[\text{the children}]_z$, because $[\text{she}]_x$ thought $[\text{they}]_z$ would like $[\text{it}]_y$.

There are three real-world *entities* mentioned in this sentence: Alice, Alice's book and the children. References to them are marked with x, y and z respectively. These references are called *mentions*. The mentions that refer to the same entity are coreferent with each other. Thus, for example, "Alice", "her", and "she" are all coreferent with each other. "Alice" is however not coreferent with "they", as they refer to different entities.

Coreference resolution - algorithmically detecting coreference - turns out to be a surprisingly difficult task. References are often ambiguous in different ways, requiring real-world or common sense knowledge. Consider the example above. In its current state it should be rather simple, since the algorithm could match the plural "the children" with "they", the singular animate "Alice" with "her" and the singular inanimate "her book" with "it". What if we replace "the children" with "Eve" instead? In this case it is still obvious to a human reader what is going on:

> $[\text{Alice}]_x$ gave $[[\text{her}]_x$ book$]_y$ to $[\text{Eve}]_z$, because $[\text{she}]_x$ thought $[\text{she}]_z$ would like $[\text{it}]_y$.

It would make no sense for Alice to give the book to Eve because Eve thought Alice would like it, so the only solution is that it was Alice who thought

Eve would like it. Encoding this type of reasoning in an algorithm is not trivial, however. Even the first example is not necessarily an easy task, as "they" can also be used as a singular gender neutral pronoun, and might thus be inferred to refer to Alice.

This thesis will focus on coreference resolution in Swedish, a task that is significantly less studied than coreference in English. In the study we implement and compare two different methods for coreference resolution, adapted from methods for coreference resolution in English. One of the methods is a deterministic method that relies on handcrafted rules that are applied in order of precision [2]. The second algorithm is an evolution of the rule-based algorithm that replaces some of the rules with random forests, a type of machine learning based classifier [3].

The results of this study could further progress in coreference resolution in Swedish, as well as progress in more high-level tasks such as information extraction and question answering. This would help researchers within natural language processing and in extension Swedish speakers in general, as they could get access to improved technology for things such as machine translation, grammar checking and other language tools that are important in the daily life of many users. The results could also be of use for other small languages, that might benefit from less resource intensive algorithms.

1.1 Definitions

1.1.1 Mention

A mention is a noun phrase or possessive that may or may not corefer with other mentions in the text. In the first example above, the mentions are: "Alice", "her", "her book", "the children", "she", "they", and "it".

1.1.2 Coreference Cluster

A coreference cluster is a set of mentions that are coreferent, or that an algorithm predicts to be coreferent. That they are coreferent means that they refer to the same real-world entity. For the more exact definition of coreference used in this thesis, see Chapter 3.

1.1.3 Anaphors and Antecedents

When discussing the linking of two potentially coreferent mentions, we will call the mention that occurs first in the text the *antecedent* and the later mention the *anaphor*. This is not strictly correct, as not all coreference is anaphoric according to the linguistic definition [4], but it is commonly used terminology in coreference resolution research.

1.1.4 Gold and Predicted

Throughout this thesis we will refer to mentions and coreference clusters as either gold or predicted. *Gold* here refers to the true mentions or coreference clusters, as decided by human annotators. *Predicted* on the other hand refers to the approximated mentions/clusters found by a mention detection or coreference resolution algorithm.

1.2 Research Question

How well does the machine learning algorithm described in [3] perform on coreference resolution in Swedish, and how does it compare to the rule-based algorithm described in [2], when evaluated on our corpus?

1.3 Goals

1. Create an annotated corpus of Swedish texts that can be used for training and evaluation of coreference algorithms.

2. Implement one rule-based and one machine learning based coreference resolution system for Swedish that can be used in further research or applications.

3. Evaluate the performance of these algorithms on the corpus, compared to each other as well as to previous work.

1.4 Delimitations

The project is limited to evaluating and comparing the selected methods, and will not attempt to determine what the best method for coreference is globally.

Further, the annotation and comparison will be done based on one view of what coreference is and should be, and will not be applicable to other definitions.

1.5 Structure of the Thesis

The thesis starts with necessary theoretical background in Chapter 2. This chapter focuses on introducing the linguistic concepts necessary to discuss coreference resolution, as well as machine learning and language technology methods that are relevant for the task. It also describes some related work, and the history of coreference resolution in English and Swedish. In Chapter 3 the annotation process and data are described in detail. The description of the method continues in Chapter 4, where the selected algorithms and details of their implementation are described. The performance of the algorithms is analyzed and described in Chapter 5. Finally, the results are discussed in Chapter 6 and conclusions and future work are presented in Chapter 7.

1.6 Examples and Translations

The algorithms in this study were adapted to Swedish and evaluated on Swedish data. As such it is often necessary to give examples in Swedish, in order to capture certain properties of the algorithm correctly. In these cases, an English translation is given below. The translation should be seen as an aid in understanding the Swedish example rather than an example in itself, as some of the properties of Swedish may not translate well. However, in some cases where the concept or algorithm behaviour can be explained accurately using only an English example, this is done instead, to make the text easier to follow.

Chapter 2

Background

This chapter will go into some of the necessary background knowledge, starting more general and then going into coreference specifics. The first section is about linguistic terminology followed by a section on machine learning and different language technology concepts. Finally we discuss the history of coreference resolution for both English and Swedish, including the algorithmic developments, different text corpora and evaluation metrics that have been used, as well as the varying definitions of what coreference means.

2.1 Linguistic Background

The development of coreference resolution techniques often relies on linguistic knowledge. This is particularly true for rule-based algorithms, which require knowledge of the target language to handcraft the rules. But it is also necessary for many of the machine learning approaches, as data typically has to be preprocessed and appropriate features extracted. This section will describe the necessary terminology as well as some properties of Swedish that are relevant for coreference resolution.

2.1.1 Part of Speech (POS)

Words can be divided into different *part of speech* (POS) categories based on their syntactic function in sentences, their meaning and how they are inflected. Some examples of part of speech categories that are relevant both for English and Swedish are nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, interjections, numerals, articles, and determiners.

Exactly which categories to use, and where to draw the lines, is not always

obvious. Many different tagsets are used in language technology, with many different granularities.

2.1.2 Lemmas

The *lemma* of a word is the dictionary form of the word, typically the word that is the least marked [5]. For instance "mouse" is the *lemmatized* form of "mice", since "mice" is the plural version of the word. Another term for lemma is headword, which is not used in this thesis on order to avoid confusion with "head word", which is used to denote the head of a phrase (see Section 2.1.11).

2.1.3 Gender

In Swedish there are two forms of gender. The first is the masculine/feminine contrast, where for instance "mor" ("mother") is a feminine word, and "far" (father) is a masculine one. This contrast is not grammatically marked in modern Swedish, except for a few special cases that have not been taken into consideration in this study. This type of gender will be referred to as *natural gender* throughout this report.

The second type of gender is the *grammatical gender*. It categorizes nouns into two groups: neuter gender (Swedish: "neutrum") and common gender (Swedish: "utrum"). The grammatical gender of the noun determines the definite form of the word and the form of any adjective, determiner or pronoun used to define or refer to the noun [6].

Detecting whether the word is in neuter or common gender can be useful for coreference resolution, as it prohibits some references. For instance in the sentence "Jag hämtade den" ("I fetched it") "den" is the common form of "it", and could as such refer back to a previously mentioned chair ("en stol", common) but not to a table ("ett bord", neuter).

2.1.4 Number

The *number* of a noun phrase can be either *singular* or *plural*. This is dependent on whether or not the phrase denotes one or multiple things. Many pronouns are inflected to mark the plural form, as well as most nouns [6].

Number distinctions are useful for coreference resolution for similar reasons as gender, since for instance a plural pronoun is unlikely to refer back to a singular noun. There are exceptions however, such as in this sentence:

[Företaget]$_x$ meddelade att [de]$_x$...
([The company]$_x$ announced that [they]$_x$...)

Here "Företaget" ("The company") is singular, but "de" ("they") are in plural. This discrepancy occurs because the company is one singular unit that consists of multiple people.

2.1.5 Definiteness

That a noun phrase is *definite* means that the listener/reader is expected to be able to uniquely identify the referent [6]. An example of a simple definite phrase is "mannen" ("the man"), and an example of an indefinite phrase is "en man" ("a man"). It can be noted from this example that while the indefinite form uses a determiner along with the lemma version of the word, the definite form inflects the word. There are however many other types of definite noun phrases, among them phrases such as "denna stol" ("this chair"), where the word is left uninflected and the definiteness is marked by the demonstrative pronoun "denna", instead [6]. Definite descriptions often require context to infer what is meant. Sometimes this context is based on previous mentions in the text, and sometimes world knowledge. For example "the president" could both be a reference to a previously mentioned president, or it could be that the audience is expected to understand which president would be relevant in this context. Definite descriptions are sometimes generic, referring to an entire class of things rather than a specific specimen [6].

2.1.6 Generic Noun Phrases

A generic noun phrase is a phrase that refers to all things that match the description, rather than a specific referent. Generic noun phrases can take many forms, both definite, indefinite, plural and singular. For example both "katten" ("cat", definite singular) and "katter" ("cat", indefinite plural) could be used to refer to the species as a whole [6].

2.1.7 Animacy

Referents can be either *animate* or *inanimate*. An animate referent is considered to be living in some sense, and equipped with feelings, wants and sense. Nouns that refer to human beings are almost always animate, while references to animals may be both animate or inanimate [6]. In Swedish anaphoric pronouns in singular have different forms depending on the animacy of the referent. For

animate referents "han" ("he"), "hon" ("she") or "hen" (gender neutral), are used, whereas for inanimate referents "den"/"det" ("it") are used. There are however some exceptions to this, concerning nouns that are inherently animate but may be referred to by inanimate pronouns. This is true particularly for some animate nouns of neuter gender, such as "vittnet" ("the witness") which could be referenced with the animate "han"/"hon" or the typically inanimate "det" [6].

In an corpus study on pronominal coreference in Swedish, it was found that the scope of pronouns (how far back in the text they can refer to) depended on the animacy of the pronouns. All pronouns had relatively small scope, but animate pronouns had a significantly larger scope than inanimate ones. [7].

2.1.8 Genitive

Genitive is a form that typically marks that one entity belongs to or is connected to another entity in some manner. In Swedish genitive is marked by an "s"-suffix. It is used for nouns, proper nouns and some pronouns [6]. Here "Bobs" is the genetive form of "Bob":

> Bobs bok
> (Bob's book)

2.1.9 Metonomy

Metonomy is the act of referring to something by mentioning another thing that is closely associated to the referent [6]. An example is the usage of Moscow as a metonym for the Russian government in this news text from CNN:

> She rejected the [Russian government]$_x$'s suggestion that it is taking reciprocal action: "We don't see this as a diplomatic tit for tat," Nauert told reporters shortly after [Moscow]$_x$ announced the move.
>
> *From "Russia expels US diplomats and shuts consulate in tit-for-tat move", by Angela Dewan, Mary Ilyushina and Sebastian Shukla, CNN. March 30, 2018.*

2.1.10 Phrases and Head Words

A *phrase* is a group of words or a single word that functions as a single unit within a grammatical hierarchy. Each phrase has a so called *head word*, that

identifies the type of the phrase. For instance a phrase which has a noun head word will be noun phrase, whereas one headed by a verb will be a verb phrase. The other words in the phrase are called *dependents*. An example of a noun phrase is "the dog", while "barked the whole night" is a verb phrase. Other types of phrases include preposition phrases, adverb phrases and adjective phrases.

A special case is that a noun phrase may have an adjective functioning as a noun as its head word. For instance "de flesta konservativa" [6]. This approximately translates to "most conservatives" but the word used is the adjective form "conservative".

2.1.11 Dependency Grammar

Dependency grammar is based on the idea that all words in a text relate to each other through directed links. In a *dependency tree*, edges go from the head to its dependents. A word may have multiple dependents, but only one head. Some examples of dependency relations are: subject (from the verb to its subject), object (from the verb to its object), case (from a noun to the preposition that introduces it).

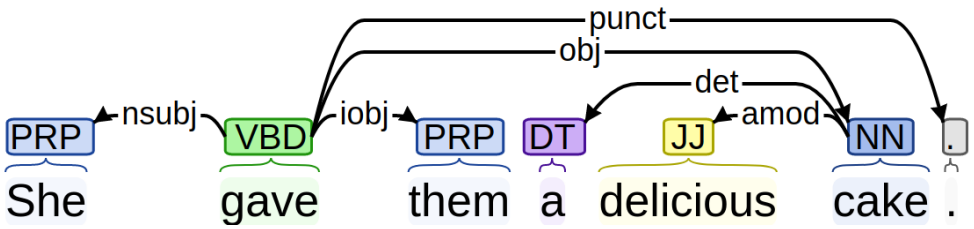

Figure 2.1: Example of a sentence with a dependency tree

In the example figure we can see a graph of the dependency grammar structure of a sentence. We see that "gave" is the root of this sentence, as it has no incoming edge. The noun phrase that acts as subject to "gave" has the head word "She", as is shown by the nsubj relation from "gave" to "She". The sentence further has an indirect object "them", which is shown by the iobj edge. The direct object is more complex. There is an edge from "gave" to "cake" with the obj type, showing that the head word of the object is "cake". But "cake" in turn has a determiner "a", and an adjectival modifier "delicious". So the object consists of the noun phrase "a delicious cake".

Universal Dependencies is a project that aims to collect and provide dependency annotated data (treebanks) for many different languages in a

unified schema [8]. This schema is used throughout this report (including the above example), and is described on the Universal Dependencies website.*

2.1.12 Predicative Expressions and Copula Verbs

A *predicative* is a part of a verb phrase that describes what the subject is or becomes. Predicatives can be noun phrases, both definite and indefinite, as well as other types of phrases [6]. Example of an indefinite nominal predicative:

> Han är en sjuksköterska.
> (He is a nurse.)

The verb used together with a predicative is sometimes called a copula verb. *Copula verbs* are verbs that carry little meaning in themselves, and are only used in this type of expressions. For Swedish this includes "vara" ("be") [6]. Sometimes the term is used for a wider set of verbs that are also used in predicative constructions, such as "bli" ("become") and "heta" ("be called"). It is this definition that has been used in our annotation guidelines.

2.1.13 Apposition

Apposition is a language construct where phrases, typically noun phrases, are placed next to each other and the appositive describes the referent of the dominant noun phrase in some way. This relation is similar to the relation between the subject referent and the subjective predicative in a predicative clause [6].

Apposition can be *restrictive* or *non-restrictive*. Non-restrictive appositives are typically surrounded by commas, but it can also be parentheses or dashes [6].

> Min syster, Alice, kommer i helgen.
> (My sister, Alice, is coming this weekend.)

The previous apposition is non-restrictive, since it does not apply any further restrictions on who the referent is. Rather, the writer is informing the reader that the name of her sister is Alice.

Restrictive apposition, on the other hand, is typically not marked with any punctuation:

* https://universaldependencies.org/u/dep/

Min syster Alice kommer i helgen.
(My sister Alice is coming this weekend.)

This example is restrictive, since the appositive ("Alice") further specifies the referent, rather than adding information. For instance the writer might have multiple sisters, and is informing the reader that it is Alice that is visiting, and not Eve.

2.2 Language Technology Concepts

2.2.1 Decision Trees and Random Forests

Decision tree learning is a form of machine learning that can be used to classify data points based on their properties. The idea is to form a tree with a question at each node. To classify a data point we start at the root and then traverse the tree, choosing which child node to go to depending on the answer to the question. Depending on which leaf node we end up in, we can then predict a label for the data point.

As an example, consider we want an algorithm that can determine whether a certain day would be good for a picnic, based on weather data. The data available to us is temperature, wind speed, whether it is raining and whether it is sunny. A *feature vector* could look something like this:

Wind Speed	Sunny	Rainy	Temperature

Our training data would consist of a set of such feature vectors, together with a label saying whether or not we thought that day was a good picnic day. Training a decision tree on this data could lead to a tree similar to the one in Figure 2.2. Consider this input vector:

7	0	0	25

If we try to predict the label for this data point using the tree in Figure 2.2, we will first note that it is not sunny, so we have to check if it is raining. It is also not raining, so we have to check if the temperature is over 20. It is, and we now predict that this day would be good for a picnic.

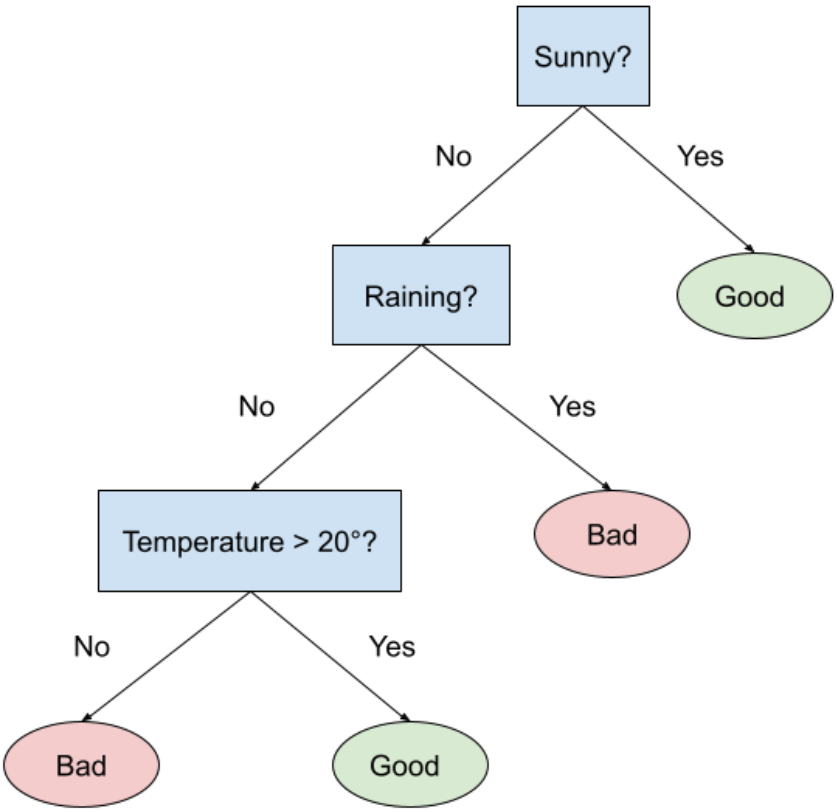

Figure 2.2: Example of a decision tree that attempts to predict whether a certain day will be good or bad for a picnic, based on weather data.

Training

To train a decision tree, we start at the root node of an empty tree. We then choose a question that will split the training dataset according in order to minimize the *Gini impurity* of the new splits.* Gini impurity is a measure of how often a randomly chosen element from a set would be given the wrong label, if we choose a label randomly according to the true label distribution of the set. This means that the Gini impurity will be high in a set with a wide and even distribution of labels, and low in sets where a few labels dominate.

Let J be the number of labels used, and the labels be numbered from 1 to J. Further, let p_i denote the fraction of the set that have label i. Then the Gini

* Some decision tree variations instead use information gain and entropy decrease.

impurity of the set is given by:

$$I_G = \sum_{i=1}^{J} p_i(1 - p_i) \tag{2.1}$$

When we have chosen a question at the root node based on Gini impurity, we split the training data according to the question, and repeat the process at the two child nodes. This process of splitting is then repeated recursively and is only stopped when the data points in a subset at a node all have the same gold label, or no question could improve the situation. It is also possible to add other stopping criteria, such as limiting the depth of the tree.

Properties

A great benefit of decision trees are that they are interpretable [9]. For each classified data point, we can inspect the path that was taken and determine exactly why this decision was made. Decision trees are also good at conjoining features, i.e. finding combinations of basic features that together give a strong signal, which has been shown to be important for coreference resolution. Further, they are good at identifying the useful features in a large feature set, and require less resources than many other machine learning methods. One issue with decision trees however, is that they tend to overfit, i.e. create overly complex trees that accommodate the training data but do not generalize well. One solution to this problem is a concept called random forests.

Random Forests

Random forests operate by training multiple decision trees, and then letting these trees vote on the classification. These trees are trained using the bootstrap aggregation, or bagging, technique. The bagging technique consists of repeatedly selecting a subset of the training data, and training a new tree using this subset. In random forests, a second randomization technique is also employed. At each split point in the trees, a random subset of the features is selected, limiting the options for the question choice. These two changes together ensure that the trees in the random forest will be diverse, and together will form a classifier that is less likely to be overfitted.

2.2.2 Word Embeddings

Word embeddings are representations of words as real-valued vectors. These vectors should be generated in such a way, that words whose vectors are close to each other in the multi-dimensional vector space, are close to each other semantically in some sense. See Figure 2.3 for an example. In the image the word vector space, which has many dimensions, has been projected onto a 2D space using an approximate PCA projection [10]. Popular techniques to create word embeddings include the word2vec algorithm [11], which uses a neural network model to create the embeddings, and the GloVe algorithm [12].

A classic experiment with word vectors is to show that "king" is to "man" as "queen" is to "woman". This is done by subtracting the word vectors as shown in the below equation:

$$\text{man} - \text{woman} \approx \text{king} - \text{queen} \tag{2.2}$$

Capturing this type of relations between words is one of the great strengths of word vectors, but it can also be a weakness. When the word vectors capture patterns in text, they also capture biases and stereotypes. Bolukbasi et al. [13] studied gender biases and possible ways to resolve them. They found that while the above relation holds, so does for instance the below relation:

$$\text{man} - \text{woman} \approx \text{computer programmer} - \text{homemaker} \tag{2.3}$$

The effects of bias in word vectors are further discussed in Section 6.4.

2.2.3 Pointwise Mutual Information

Pointwise mutual information (PMI) is a measure of how the actual probability of two events co-occurring compares to what we would expect it to be based on the individual probabilites and an assumption of independence. This measure is often used in computational linguistics, for example to measure the co-occurrence of certain words [14]. PMI is given by the following equation, where x and y are the two events:

$$pmi(x, y) = \ln \frac{p(x, y)}{p(x) * p(y)} \tag{2.4}$$

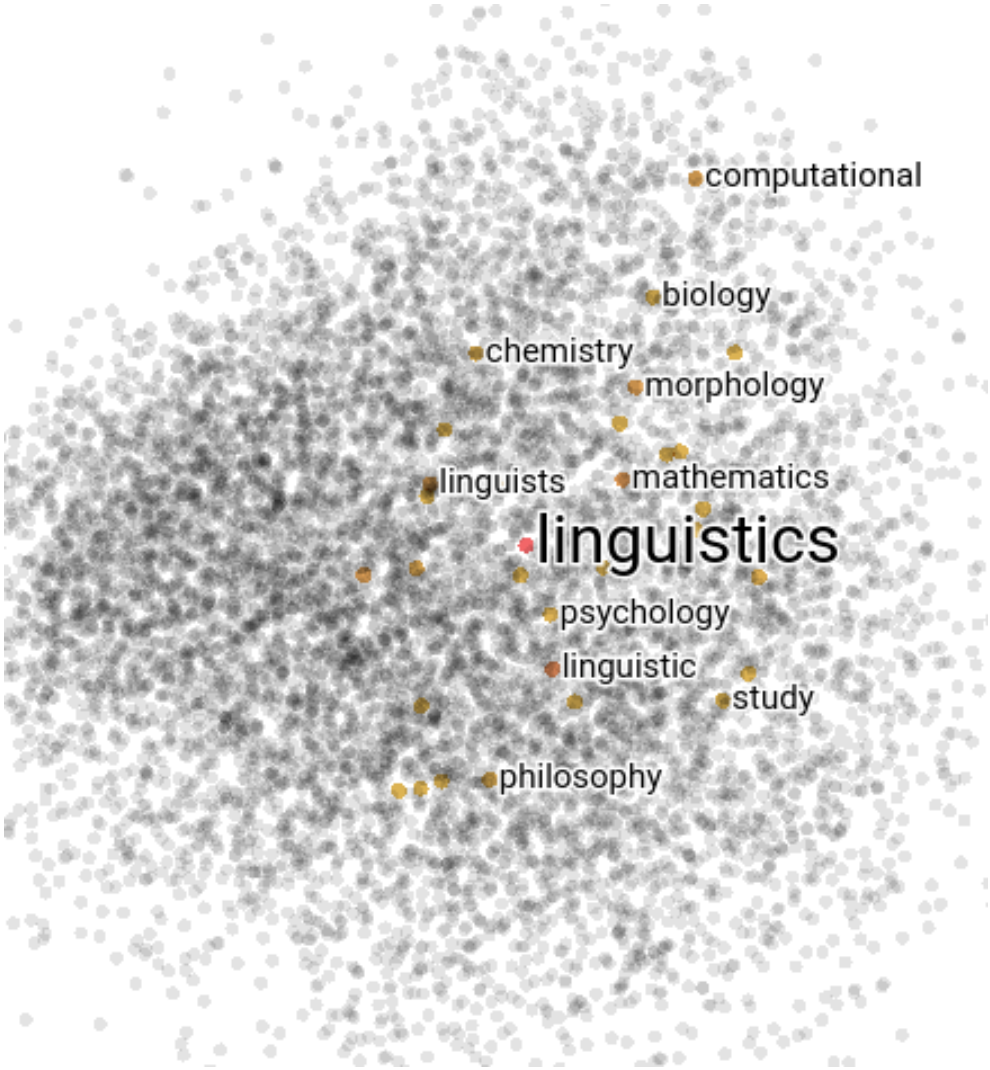

Figure 2.3: Word vector space illustration showing some of the most related words to "linguistics", as determined by their distance from the "linguistics" point in the original space. The visualization was created using the example word vector set in the TensorFlow projector tool (https://projector.tensorflow.org/). The visualization was created using an approximate PCA projection [10] and the 10K word2vec vectors.

2.3 Coreference Resolution

Now that we have explained the necessary background, we will move on to the subject of coreference resolution and examine the definitions, history and previous work on algorithms, both for English and Swedish.

2.3.1 Coreference Terminology and Theory

Two of the most important concepts when discussing coreference resolution are mentions and coreference clusters. A mention is a phrase in a text that refers to some real-world entity, that could be abstract or tangible. Examples of mentions are "the horse", "an old tree", "he" and "algebra". A coreference cluster (or coreference chain) is a set of mentions that corefer, i.e. refer to the same entity.

The exact definitions of mentions and coreference have varied between different studies and between datasets. As we go through the history of coreference resolution research we will note these differences in point-of-view. For the definitions used in this study, see Chapter 3.

2.3.2 History, Datasets and Definitions

Evaluating a coreference resolution algorithm requires an annotated dataset, where humans have marked out correct mentions and coreference clusters in texts. Machine learning algorithms also need such texts for training, making the data requirements even greater. Since such annotation efforts are expensive, there are only a few large datasets that have been used for English coreference research, and their definitions of what coreference is have shaped the direction of the research.

MUC

The Message Understanding Conferences (MUC) began in 1987 as a project to foster research in automatic understanding of military messages, and consisted of conferences as well as coordinated evaluation on set tasks [15]. The first MUC corpora that included coreference, MUC-6 and MUC-7, were created and released in the late 1990s. This effort was important both as it was the first substantial corpora, but also because it came to define much of the limitations on coreference. For instance, things like bridging (part-whole relations, such as connecting a reference to a car to a reference to its wheels) were explicitly excluded [16]. In conjunction with this, a standard scoring

method for coreference resolution was also developed, which was called the MUC score (see 2.3.3 MUC).

ACE

The next big development came in the years 2000-2004 when the five ACE corpuses were released. They were larger corpora, and for instance ACE04 consisted of 350k words in the English dataset. The ACE task also provided data for Chinese and Arabic, in addition to English. One important difference from the MUC corpus was that ACE focused on a more restricted version of coreference, where coreference clusters only needed to be identified for entities belonging to one of the ACE entity types (e.g. PERSON, ORGANIZATION, LOCATION) [17]. For the ACE task an ACE metric was developed, but it is no longer commonly used.

OntoNotes and the CoNLL-2011/2012 Shared Task

The most recent major corpus that have been released is the OntoNotes dataset. OntoNotes contains coreference annotations for texts in English, Arabic and Chinese. Using the OntoNotes dataset, two shared tasks were organized within CoNLL (Conference on Computational Natural Language Learning), where different research groups attempted to find the best coreference resolution algorithms, using the same train and test data from OntoNotes. The 2011 version of the shared task only involved the English portion of the data, while the 2012 version opened up for competition in all three languages. These tasks also introduced the CoNLL score for coreference resolution evaluation, which is described in 2.3.3 CoNLL [18].

OntoNotes set a new direction for coreference resolution, as it has a wider definition than the one in the ACE corpus. All noun phrases, pronouns and heads of verb phrases are considered potential mentions. Two types of coreference are marked, identity (IDENT) and appositive (APPOS). Identity coreference is used for links between pronominal, nominal, and named mentions of specific referents. Appositives have their own marker due to being seen as attributive rather than coreferential (i.e. they provide extra information about a referent rather than being their own separate reference). Some interesting guidelines from OntoNotes are:

- Generic "you" is not marked.
- Generic nominal mentions can be linked with pronouns referring to them, and with definite mentions, but not with other generic mentions.

- Attributes signaled through copular structures are not marked as coreferential.

The level of inter-annotator agreement was studied for the OntoNotes corpus. This showed an agreement of 80.9% for English newswire text. Agreement for other English text types varied between 78.4% and 89.4%. The most common disagreement types were also analyzed. The most common category was Annotator Error, a catch-all category for errors that did not fit any other description. This was closely followed by Genuine Ambiguity, where the text simply was ambiguous, for instance pronouns that could refer to multiple things. These two groups contributed around 25% of the errors each. Finally, Generics contributed around 10% of the errors. This type of error meant that one annotator considered a mention generic, and the other not. The rest of the errors came from various smaller categories [18].

2.3.3 Evaluation Metrics

Many different evaluation metrics for coreference resolution have been developed throughout the years. In this section we will briefly describe those that were used to score our algorithms. For every metric except the CoNLL score we will give the equations for precision and recall. A combination of precision and recall can then be calculated in the form of an F1-score, according to the following equation:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.5}$$

MUC

The MUC metric views a coreference cluster as a set of linked mentions, where each mention is linked to at most two others. The metric is intended to quantify how many link modifications need to be done to turn the predicted clustering into the gold clustering. Precision and recall are calculated according to the below equations [19]. T is the set of gold clusters, and R is the set of predicted clusters. The $partition$ function takes a cluster c, and a set of clusters S and returns all clusters in S that have some overlap with c.

$$Precision(T, R) = \sum_{r \in R} \frac{|r| - |partition(r, T)|}{|r| - 1} \tag{2.6}$$

$$Recall(T, R) = \sum_{t \in T} \frac{|t| - |partition(t, R)|}{|t| - 1} \tag{2.7}$$

B^3

B^3 is a metric that was proposed in 1998 as a response to the MUC metric [20]. The idea is to compute precision and recall for each mention, with regards to how many of the other mentions in their predicted cluster are correct. The scores for each mention are then combined.

In the below equations the mentions are numbered from 1 to N. w_i is the weight of mention i. This value is typically set to $\frac{1}{N}$ in order to weigh all mentions equally, but could also be distributed some other way [20].

$$Precision_i = \frac{\text{number of correct elements in the output cluster containing mention i}}{\text{number of elements in the output cluster containing mention i}} \tag{2.8}$$

$$Recall_i = \frac{\text{number of correct elements in the output cluster containing mention i}}{\text{number of elements in the gold cluster containing mention i}} \tag{2.9}$$

$$FinalPrecision = \sum_{i=1}^{N} w_i * Precision_i \tag{2.10}$$

$$FinalRecall = \sum_{i=1}^{N} w_i * Recall_i \tag{2.11}$$

CEAF

CEAF (Constrained Entity Alignment F-Measure) was proposed in an article in 2005, in two versions: entity-based and mention-based [21]. We will here explain the entity based version, as that is the one we will use.

We first define ϕ, which is a function that takes a gold cluster ("true cluster") and a predicted cluster ("result cluster") and calculates the similarity.

$$\phi(T_i, R_j) = \frac{2 * |T_i \cap R_j|}{|T_i| + |R_j|} \tag{2.12}$$

We then calculate precision and recall by finding the one-to-one mapping between predicted clusters and gold clusters that generates the highest scores, as defined in the below equations [4][21]. $m(r)$ is a function that takes a predicted cluster r and returns the gold cluster that r is mapped to in this mapping. T is the entire set of gold clusters, and R the set of predicted clusters.

$$Precision(T, R) = max_m \frac{\sum_{r \in R} \phi(r, m(r))}{|R|} \tag{2.13}$$

$$Recall(T, R) = max_m \frac{\sum_{r \in R} \phi(r, m(r))}{|T|} \tag{2.14}$$

CoNLL

The CoNLL score is calculated as the unweighted average of the F1 scores for MUC, B^3 and entity based CEAF. [18].

$$CoNLL = \frac{B^3 + MUC + CEAF_E}{3} \tag{2.15}$$

2.3.4 Coreference Resolution Algorithms for English

The historical development of coreference resolution algorithms have largely been centered around coreference resolution for English. In this section we will discuss the developments throughout the years, and mention some important strategies and algorithms. The algorithms can be coarsely split into rule-based algorithms, and algorithms that are data driven and use machine learning. We will start by discussing rule-based algorithms, followed by an overview of machine learning approaches. Finally we give a more detailed description of the two algorithms that we have based our own work upon.

Overview of Rule-Based Algorithms

For a long time, rule-based algorithms were the state-of-the-art for coreference resolution. The type of approach and the philosophy behind them have varied, from resource-heavy approaches with complex rules, to more light-weight systems. Many of the algorithms in the beginning focused on the simpler problem of anaphora resolution, rather than coreference resolution. Anaphora resolution seeks to resolve references where one mention relies on an earlier mention for the identification of the referent. This includes for example

pronouns that refer back to a proper noun, but not the repetition of the same proper noun, even though they have the same referent [4]. Early influential works on anaphora resolution include Hobbs' naïve algorithm from 1978 [22] and the algorithm presented by Lappin and Leass in 1994 [23].

As machine learning approaches started taking over in the past two decades, the focus of rule-based algorithms shifted to using simpler rules and ordering these rules according to precision. Influential algorithms in this later development include the Haghighi and Klein algorithm from 2009 [24] and the sieve method from 2010 [25]. Another interesting contribution is the work by Zeldes and Zhang in 2016 [26], which was published much later than most other rule- based algorithms. Their algorithm was an attempt to address some issues with the current state of coreference research. The current ML approaches rely on large amounts of training data, which is difficult to produce for low-resource languages. Additionally, it is hard to retrain the algorithms for other definitions of coreference, making the progress very focused on the definition used in the OntoNotes corpus. Their approach to address this was to create a coreference system that is entirely rule-based, has a high modularity and that can be easily configured [26].

Overview of Machine Learning Algorithms

During the past decade, machine learning approaches for coreference resolution have been growing in popularity and are now the state-of-the-art. Most non-neural ML approaches can be split roughly into three groups: mention-pair models, entity-mention models and mention-ranking models.

Mention-pair models work by training a binary classifier that determines whether or not two mentions are coreferent. All these classifications are then used to form clusters in some way. The mention-pair models can differ in three important ways: selection of training examples, classification method and clustering method. One of the earliest methods of selecting training examples is the one by Soon et al. in 2001 [27]. For each mention, the closest coreferent antecedent is chosen as positive example, and any non-coreferent mentions that come between them are chosen as negative examples. This method has then been modified in different ways, for instance to avoid training examples that would be hard for a human [4]. When it comes to the classification part, decision trees and random forests have been popular choices for classifiers, used for example in [28] and in [3]. Memory based learners, statistical learners and systems that learn rules have also been widely popular [4]. Finally, the clustering methods can differ. Some, for instance [27], use a closest-first

approach where the closest antecedent that is marked as coreferent by the classifier is chosen. Other systems have used more complex approaches, such as in [29] where a graph partitioning algorithm is used for the clustering phase.

Mention-ranking models improve on mention-pair models by instead training a model that produces a ranking of all the antecedents. Two prominent algorithms that used this approach are [30] and [31].

Entity-mention models attempt to solve another issue with the mention-pair approach. Viewing the problem as a question of classifying independent links risks that the classification does not adhere to transitivity. A might be classified as coreferent with B and C, while B and C are classified as not coreferent. Further, it prohibits information sharing. If a cluster already includes a female pronoun for example, we probably do not want to add a male pronoun to it, even though the candidate antecedent for this pronoun is gender neutral. In entity-mention models the classifier is trained to determine whether a certain mention should be coreferent with a partially formed cluster, rather than with a single mention. This allows usage of cluster-level features. Examples of successful entity-mention models include [32] and [33].

Apart from these types of models, there are also more unique approaches, such as [34], which views the clusters as trees, and trains a perceptron to detect latent tree structures. Finally, the currently best-performing category is neural algorithms. One example of a successful algorithm in this category is the end-to-end neural approach presented in [35].

The Stanford Sieve Methods

Among the earlier mentioned methods, one of the most successful ones was the sieve-based method presented in three papers: [25], [36] and [2]. The method is entirely rule-based, and consists of a set of rules (called *sieves*) that are applied in order of expected precision. The sieves gradually cluster mentions together, attempting to find antecedents to the first occurring mention in each cluster. This method performed well in the ConLL 2011 Shared task and is used in the Stanford NLP library (alongside later neural approaches).

A machine-learning based development of this approach is presented by Lee et al. in 2017 [3]. They performed an error analysis of the rule-based sieve method, and then improved it by replacing some of the rule-based sieves with sieves that use random forests. The results were comparable to [37], and significantly better than the purely rule-based version of the sieve method.

This report uses adaptions of these algorithms to solve coreference resolution for Swedish. Details of how the algorithms work, and how we have modified

them, can be found in Chapter 4. These algorithms were chosen for our task for multiple reasons. They are both high-performing resolution algorithms, that have shown good results on the standard datasets for English. Additionally, the fact that the machine learning algorithm is based upon the rule-based approach makes the comparison easier and more useful. The modularity of the system also makes it easy to adapt to a new language, as it is easy to discard and add sieves to fit the new requirements. Random forest learning is also an appealing choice for a machine learning model, since they perform reasonably well on small amounts of data and are more interpretable than many other types of models.

2.3.5 Swedish Coreference

Hybrid Method for Coreference Resolution

A major previous work on coreference resolution in Swedish is the thesis by Nilsson in 2010 [38]. They performed annotation of a corpus according to the Norwegian BREDT schema, and then evaluated a hybrid method for coreference on this corpus.

The corpus consisted of financial texts from the internet, and contained a total of 66 texts annotated with coreference relations. The annotation schema differentiated between different types of relations, such as coreference, predicatives, subset relations and metonomy. Analysis showed that 85% of annotated relations were coreference, with predicatives (3.9%), subset (3.8%) and superset (3.5%) coming next.

The implemented coreference resolution algorithm combines machine learning with rules for selecting the possible antecedents. These rules are primarily based on accessibility theory, which ranks different noun phrases based on how "accessible" they are in a text. For pronouns for instance, animate pronouns are allowed a longer scope than inanimate ones. See [39] for more information about accessibility theory.

For the selected antecedents, each antecedent is classified as coreferent or non-coreferent by a k-nearest-neighbor algorithm. Essentially, each antecedent-anaphor pair in the training data is represented by a feature vector, containing features describing the anaphor, the antecedent and their relations. When a new pair is to be classified, the vector for the pair is calculated and the labels of the closest training vectors in the vector space determine the predicted label of this vector. The model uses a large amount of features, some of which have been include in our own work.

The clustering approach used is the so called aggressive-merge clustering. This approach merges the anaphor with all antecedents that the classifier identifies as coreferent (rather than, for example, only the closest one). This method is supposed to favor recall, likely at the expense of precision [38].

The algorithm was evaluated using gold mentions and achieved the following MUC scores. Precision: 67.56 Recall: 66.52 F1: 67.04 [38].

SUC-CORE

SUC-CORE is a Swedish corpus annotated with coreference, based on the texts in the Stockholm-Umeå Corpus (SUC). The text types are both informational, such as texts from news and textbooks, and imaginative, in the form of excerpts from novels [1].

The report describing the corpus includes some interesting data on how likely mentions of different types are to be the first mention in a cluster (initial), versus occurring later in the cluster (subsequent). An example of these statistics can be seen in Table 2.1.

Table 2.1: Frequency of cluster positions for different mention types in the informative prose part of SUC-CORE (as percentages) [1].

	Single	Initial	Subsequent
Common nouns	30.2	23.1	46.7
Proper nouns	76.5	10.0	13.5
Pronouns	11.4	3.9	84.7

SUC-CORE uses a different set of annotation rules than the BREDT-schema used in [38]. Only noun phrases (and possessives) are marked as mentions. The only relation that is annotated is coreference. These are some of the more interesting annotation rules in SUC-CORE [1]:

- A substitution test is used to check for coreference. If two mentions can not switch places without the meaning changing, they are not coreferent. This means, among other things, that bridging (part-whole relations) and bound anaphora are excluded.
- Metonomy is annotated as coreference.
- Generic mentions are annotated, but clusters of generic mentions are carefully separated from non-generic clusters.
- The generic "man" ("one", in English) is only annotated when it has a clear referent.

- Function type expressions are excluded, so phrases like "the number of voters" can not corefer.
- Plural mentions with coordinated antecedents are annotated, but not mentions with split antecedents (such as two people being mentioned in different parts of a text, and later being referred to as a couple).

Chapter 3

Data and Annotation

A requirement for this project was to have a dataset manually annotated with coreference clusters. This was necessary both for the purpose of training the machine learning method, and for evaluating both methods. The only previously available corpus with this type of annotation was the SUC-CORE corpus, and as this corpus is rather small we decided to create our own. The following sections describe the texts selected for annotation, the rules guiding the annotation, and the annotation process itself.

3.1 Data

The texts that were annotated come from the Swedish corpus Talbanken. Talbanken is a database consisting of roughly 6000 sentences and 95000 tokens. The sources include textbooks, information brochures and newspaper articles. The corpus has an official training/development/test split. We used the test division as is, which is important since some of the tools we use have been trained on Talbanken and used this division. However, we have made our own split of train and development. The sizes of the three datasets we used can be seen in Table 3.1.

Table 3.1: Size of the dataset.

	Documents	Mentions	Coreference Clusters
Training	24	1712	453
Development	7	494	135
Test	14	1242	353
Total	45	3448	941

3.2 Annotation

Annotating coreference data is a non-trivial task, since there are many corner cases and different possible interpretations of what coreference is. There were three main concerns when deciding on annotation rules:

1. Linguistic interpretability: How well do the rules correspond with our knowledge of language? A consistency and logic in the rules might help machine learning algorithms learn.

2. Ease of annotation: How easy is it for the human annotators to follow these rules? Both in terms of speed of annotation, and minimizing differences between different annotators.

3. Richness of information: How useful is the information for task it will be used for?

In particular concern 1 and 3 are often at odds with each other, as a multitude of different connections might be useful for e.g. an information extraction system, even though they are very different linguistic constructs. 1 and 2 on the other hand, often go together. A rule set that is less linguistically consistent often lead to more complicated rules, as various corner cases need to be considered, and make it harder for annotators to understand and consistently annotate.

One issue with point 3 is to define what the task is. In our case we are primarily developing coreference algorithms to be used for generation of reading comprehension questions, but for such a resource intensive annotation project it would be optimal if the annotated dataset can be used for coreference for other tasks. Another factor when it comes to point 3 is whether the information would be possible to extract in some other way. For instance apposition and copula constructions follow clear patterns, and might be

possible to extract by using some simple rule-based system. It would also be comparatively easy to make a second annotation pass to add all these links, if we changed our minds. This makes these constructions less important to include in the annotation.

3.2.1 Annotation Rules

We will now discuss some of the more difficult distinctions in the annotation rules, as well as ones that differ from the standard choice. For the complete set of guidelines used during annotation, see Appendix A.

What kind of relations should be marked?

Only coreference should be marked, i.e. mentions that have identical referents. We further exclude apposition and connections through copula constructions.

What is eligible to be a mention?

A mention is either a noun phrase or a possessive pronoun. Verb phrases and larger stretches of discourse, such as sentences, can not be marked as mentions. This choice was made partly to make the dataset more compatible with SUC-CORE, which also used this rule [1]. The other reason is that it speeds up annotation, as coreference for events, paragraphs etc are usually harder to identify.

The below sentences has an example of coreference that should not be marked. The first marked phrase is a verb phrase, and is thus not eligible to be a mention, even though the later noun phrase does refer back to it.

> Alice [åkte till Indien under sommaren]. [Denna resa]...
> (Alice [went to India during the summer]. [This trip]...)

An example of a larger stretch of discourse could be an entire paragraph of arguments, that is then referred back to with a noun phrase such as "the previously mentioned reasons".

Finally, only mentions that corefer with at least one other mention should be marked. This decision is only made to increase annotation speed, as the decision on what should be a mention is harder to make if you have to consider things that have no references to them. If singleton mentions are needed in the future, it is easy to augment the annotations with these by making another annotation pass. On this point, our annotations differ from SUC-CORE [1]. To use the corpuses together however, it should be easy to ignore marked singleton mentions in the other corpus.

What is included in the mention?

Restrictive appositions should be included in the mention, and can be identified by the lack of comma. Preposition phrases are included if they are restrictive modifiers. For example in "Enligt Utskottet för ekonomisk politik..." ("According to the Commission for Economic Policy...") it is not enough to include "Utskottet" ("the Commission") to identity the referent, and as such the mention must be "Utskottet för ekonomisk politik" ("the Commission for Economic Policy").

Function type expressions

Function type expressions can be mentions, but their values can not. For instance in this example the marked mentions are coreferent, but not the unmarked numbers:

> [Antalet röster]$_x$ var 2700. Förra året var [antalet]$_x$ 3000.
> ([The number of votes]$_x$ was 2700. Last year [the number]$_x$ was 3000.)

This is different from SUC-CORE, where function-type expressions are excluded from coreference [1].

3.2.2 Annotation Process

The texts were annotated by the author and two other annotators. The first three texts were annotated by all annotators, to discuss and ensure uniformity of annotation. For the rest of the texts only one annotator was assigned to each text.

The annotation was done in the Textinator tool, developed by Dmytro Kalpakchi at KTH. Textinator is a visual tool that allows marking mentions in a text, and linking them together into coreference clusters.

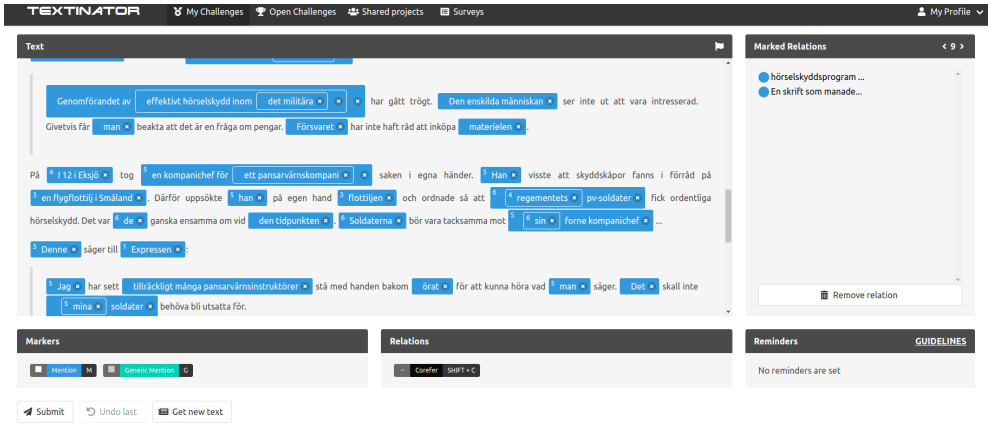

Figure 3.1: Annotation in progress in Textinator

Chapter 4

Algorithms and Implementation

This chapter describes the algorithms we compared and their implementation. It starts with a description of the mention detection module, and then moves on to first the rule-based algorithm and then the machine learning algorithm. These sections describe the details of the algorithms and compare them to the original algorithms for English they were based on, motivating the changes that were made. Finally an overview is given of the implemented system, as well as descriptions of all the libraries, technologies and tools used.

4.1 Mention Detection

In order to get a complete coreference resolution pipeline that can handle raw text, it is necessary to implement a mention detection step. This mention detection algorithm is used both for the rule-based coreference algorithm, and for the machine learning algorithm.

The mention detection algorithm is inspired by the method for mention detection using dependency trees that is presented in [3], but has been extended and modified. The algorithm works by identifying words that might be head words of mentions, and then using the extracted dependency trees to identify the full extent of these mentions. Below follows a description of the rules of these two steps.

4.1.1 Head Word Selection

All words that are marked as pronouns, proper names or nouns are considered as potential heads for mentions. No other words are considered, since we focus only on coreference resolution of noun phrases. Not all word with these POS-

tags are suitable, however, and some more filtering can be done by examining the dependency relation the word has to its head. The following dependency relations exclude the word:

- expl: Marks the word as an expletive. In Swedish this captures the existential "det", as in "det finns flera lösningar".
- det: Marks the word as a determiner, for instance "den" i "den vackra trädgården" ("the beautiful garden"). It is the word it determines, which will be the head word of the mention.
- fixed: Marks the word as part of a fixed expression. For example in "på grund av", "grund" has the fixed relation.
- nummod: Marks a numeric modifier. For example in "40 euro", "40" serves as a numeric modifier, and is not eligible for coreference on its own.
- compound: Marks the second part of a compound. In Swedish this is only used for foreign expressions, as Swedish compound words should be written as one.
- flat:name: This relation marks that the token is part of a name, and has an earlier part of the name as its head. In this case the first token of the name should be the head of the mention instead. For example in "Isaac Newton", "Newton" would have the flat:name relation.
- appos: Marks the word as being in an appositive relation. This rule differs from the others in that it is very dataset dependent. We chose not to annotate apposition as coreference, and thus appositives are not eligible mentions, as any future references back should be connected to the head of the appositive instead. Thus disabling this filter would be suitable for datasets that does not have this restriction.

For more examples and more extensive descriptions, see the Universal Dependencies website*.

* https://universaldependencies.org/u/dep/

4.1.2 Generating Phrases from Head Nodes

For each selected head word, we now have to find the extent of the mention they are head of. The first step is to find the subtree that the head word is at the root of. This is done by following the edges from the head word until we reach leaf nodes, including every word on the way. Edges with dependency relation type "orphan" and "appos" are not followed.

From this string multiple extents are generated. The full string is always included. If the string contains any copula constructions, the part of the string that comes after the copula verb is added. As an example, for the full string "de som är från Kalifornien" ("those who are from California"), we will also add "från Kalifornien" as a mention string. If the string contains "och" ("and"), the part that comes before "och" is also added.

These strings are then trimmed. Tokens are removed from the beginning until the dependency relation of the first token is not one of 'case', 'cc', 'punct' or 'mark'. This will turn "från Kalifornien" into only "Kalifornien". Any punctuation that is not quotation marks is removed from the end of the string. If the string contains a comma at any point, everything from the comma and onward is removed.

4.1.3 Example

Given the example sentence in the dependency tree illustration in Figure 4.1, the head word detection algorithm would detect two pronoun head words, "she" and "them", and one noun head word, "cake". The pronouns have no dependents, and are thus added to the list of mentions as they are. The word "cake" however, has some dependents, so we follow these connections to get the complete extent of the mention, which is "a delicious cake".

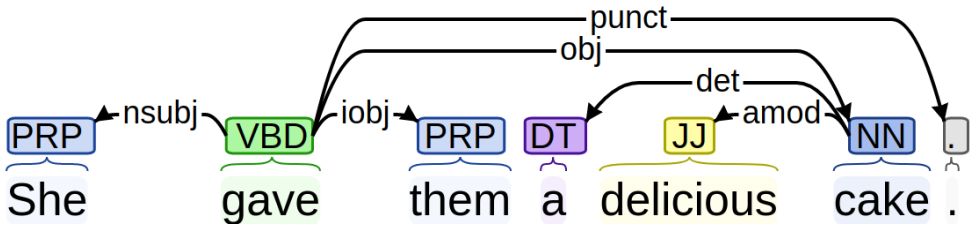

Figure 4.1: Example of a sentence with a dependency tree

4.2 Rule-based Algorithm

The rule-based coreference prediction algorithm is based on the algorithm presented in [2], but slightly simplified and modified for Swedish.

The basic idea is that we consider a clustering problem, where all mentions that are coreferent with each other should end up in the same cluster. At the beginning of the algorithm, every mention is in its own individual cluster, and as the algorithm progresses we will merge clusters with each other as coreference is detected. The steps of the algorithm are called *sieves* (the name coming from a long tradition within coreference research). Each sieve is a boolean function that considers two mentions and decides if the clusters they belong to should be merged. As an example, we could have a simple sieve that will merge the clusters if the two mentions have identical strings. If we then have two clusters ("Bob", "he") and ("Bob", "a man") and applied this sieve to "Bob" (from the first cluster) and "Bob" (from the second cluster) the sieve would return TRUE, and we would merge the clusters to get ("Bob", "he", "Bob", "a man"). These four mentions will now all be considered coreferent.

The sieves we use are applied in a specific order, as described in the next section. For each sieve the following process is followed: We consider all the mentions in the text that are currently first in their clusters, meaning that no other mention in that clusters has an earlier position in the document. For each such mention, consider every candidate antecedent (mention that comes before it in the text), sorted in a specific order. Candidate antecedents that are in the same sentence as the mention are ordered such that subjects comes first. All other mentions are ordered by distance, so that the algorithm will examine close candidate antecedents before ones further away.

The first time the sieve returns true for a candidate antecedent, that candidate antecedent is selected and the mention and candidate antecedent clusters are merged. The algorithm now moves on to finding antecedents for another mention. When all mentions have been processed, the algorithm goes to the next sieve. As a variation to the original algorithm, we also experimented with only looking for antecedents for those mentions that are not marked as indefinite, see Chapter 5 for a comparison between these two versions.

4.2.1 Sieves

The following sieves are used in the algorithm, in this order. The order is chosen to make higher precision decisions first (i.e. perform the merges we

Algorithm 1: Rule-based Algorithm

```
Input: document, sieves, predictedMentions (from gold mentions or
       mention prediction pipeline)
Output: predicted clusters for document
Put every mention in its own cluster;
for sieve in sieves do
    for mention in predictedMentions do
        if mention is not first in its cluster then
            continue;
        end
        for antecedent in GetCandidateAntecedents(mention,
         predictedMentions) do
            if sieve.isMatch(mention, antecedent) then
                MergeClusters(mention, antecedent);
                break;
            end
        end
    end
end
```

Algorithm 2: GetCandidateAntecedents

```
Input: mention, predictedMentions
Output: candidate antecedents for mention
for otherMention in predictedMentions do
    if otherMention comes before mention then
        if mention.sentenceId = otherMention.sentenceId then
            sameSentenceAntecedents.append(otherMention);
        end
        else
            otherSentenceAntecedents.append(otherMention);
        end
    end
end
sort otherSentenceAntecedents in ascending order of distance from
 mention;
sort sameSentenceAntecedents with subjects first;
return(sameSentenceAntecedents + otherSentenceAntecedents);
```

are most confident in before ones that are more likely to be wrong). All sieves except for the pronoun resolution sieve require both mentions to be non-pronouns.

Exact Match

This sieve matches two mentions if their strings match perfectly (disregarding upper case/lower case).

Genitive Match

This sieve matches mention that are identical except one of them has an extra "s" at the end. For example "Bob" and "Bobs" would be a match for this sieve. This is a new sieve for the Swedish algorithm, since in English, a word and its genitive 's are separated by punctuation, which causes them to be tokenized into different tokens, and allows this case to be caught by the exact match sieve.

Lemmatized Head Word Match

An important difference between English and Swedish, is that in English the definite and indefinite versions of a noun will be identical: "the house" vs "a house". In Swedish the corresponding phrases are: "huset" and "ett hus". This means that an exact head word match will fail. For this reason, we added a sieve that checks if the lemmatized forms of the head words match. To avoid matching mentions with modifiers distinguishing them, mentions are only considered if they contain no nominals or adjectives except for the head word. So for instance "det röda huset" (the red house) would not match "ett hus" (a house) as the first mention contains an adjective.

Relaxed String Matching

This sieve matches mentions that are identical up to and including the head word.

Pronoun Resolution

The final step is to deal with pronoun resolution. This sieve requires the anaphor mention to be a pronoun, whereas the antecedent may be of any type.

The following requirements must hold for the pronoun resolution sieve to merge the mentions:

- Number agreement
- Natural gender agreement
- Person agreement
- Animacy agreement
- Grammatical gender agreement
- Sentence distance at most 3

Agreement in number, natural gender, person and animacy are all calculated on a cluster level, meaning that when considering two mentions we also examine the properties of the clusters they belong to. The complete set of tags are computed for each cluster. For example a cluster consisting of "the group" and "the students" might have number = (plural, singular), while the cluster consisting of "it" has number = (singular). If one or both sets are empty the match is allowed (since we do not know if they disagree). Otherwise, the intersection of the sets must be non-empty. In the previous example the match would be allowed, since both sets have the "singular" label. However the cluster containing "she" and "Sam" could not match with "he", since the first would have naturalGender = (female) and the second naturalGender = (male), which have no overlapping labels.

4.2.2 Example

We will now perform a walk-through of the algorithm, using the below example text. The text is slightly unnatural, which is necessary to fit many different types of references within only a couple of sentences.

> Alice visade stolt sin tavla för Bob. Tavlan var färgglad, och Bob kunde tydligt se Alices konstnärliga förmåga.
>
> (Alice proudly showed her painting to Bob. The painting was colorful, and Bob could clearly see Alice's artistic ability.)

First, the mention detection algorithm is run. It would identify the bolded words below as head words, and then use the dependency tree to identify the complete mention spans:

> [**Alice**] visade stolt [[**sin**] **tavla**] för [**Bob**]. [**Tavlan**] var färgglad, och [**Bob**] kunde tydligt se [[**Alices**] konstnärliga **förmåga**].

Now, we will start the rule-based coreference resolution. At the beginning every mention is in its own cluster. The first sieve to be used is the exact match sieve, which will cluster the two mentions of Bob:

> [Alice] visade stolt [[sin] tavla] för [Bob]$_x$. [Tavlan] var färgglad, och [Bob]$_x$ kunde tydligt se [[Alices] konstnärliga förmåga].

Next we have the genitive sieve, which will cluster Alice and Alices together:

> [Alice]$_y$ visade stolt [[sin] tavla] för [Bob]$_x$. [Tavlan] var färgglad, och [Bob]$_x$ kunde tydligt se [[Alices]$_y$ konstnärliga förmåga].

The relaxed string match sieve does not merge any clusters. Then we have the Lemmatized Head Word Match sieve. This will cluster "sin tavla" ("her painting") and "tavlan" ("the painting"), since the lemma of both of these head words is "tavla".

> [Alice]$_y$ visade stolt [[sin] tavla]$_z$ för [Bob]$_x$. [Tavlan]$_z$ var färgglad, och [Bob]$_x$ kunde tydligt se [[Alices]$_y$ konstnärliga förmåga].

Finally, the pronoun resolution sieve is run. Since "sin" is close to the mention "Alice", and there is no disagreement in number, person etc. these two will be clustered together by this sieve. Since "Alice" is already clustered with "Alices", we now have a cluster with three mentions in it:

> [Alice]$_y$ visade stolt [[sin]$_y$ tavla]$_z$ för [Bob]$_x$. [Tavlan]$_z$ var färgglad, och [Bob]$_x$ kunde tydligt se [[Alices]$_y$ konstnärliga förmåga].

This is the final sieve, and thus these are the final clusters, leaving "Alices konstnärliga förmåga" ("Alice's artistic ability") in a singleton cluster.

4.2.3 Differences From the Original Algorithm

This section will describe what changes were made compared to the original algorithm, and motivate these choices.

Sieves

To begin with, we will discuss the sieves used in the original algorithm that were not used in our implementation.

- Discourse processing: The discourse processing sieve detects speakers and links for instance the pronoun "I" within a quoted segment to the speaker of the segment. This sieve is fairly complex to implement, and was not implemented due to time constraints and the very small amounts of dialog in our dataset.

- Precise constructs: The precise construct sieve consists of several constructions, including appositive constructs and copula construction. The appositive construct is used to detect coreference, but removed at the last post-processing stage since the CoNLL standard does not accept appositive constructs as coreference. We investigated doing the same, but initial experimentation showed poor results. As an example, here are all the generated pairs from one of the texts:

  - väsentlig information (relevant information) - exempelvis TV-program (for example TV shows)
  - de jobbare som måste arbeta i olika skift (the workers who have to work different shifts) - exempelvis en veckas dagjobb (for example a week of working during the day)
  - högre kompensationsbehov (greater need of compensation) - sömnskuld (sleep debt)

  For copula constructions the quality was also low. Another part of the precise construct sieve is acronym detection. Acronym detection is slightly more complicated in Swedish than in English due to the prevalence of compound words. It would however, be a good future addition.

- Various head word sieves: The original algorithm included multiple variations on the head word sieve, putting different constraints on the mentions to match. These yielded little improvement in their experiments, and were thus not implemented for our algorithm in order to save time.

Features in the Pronoun Resolution Sieve

Our pronoun resolution sieve is slightly less advanced than the original. They use some static lexicons for number, animacy and gender detection, which we did not use. It should also be noted that the gender agreement constraint was implemented somewhat differently. We only require agreement of natural gender (masculine/feminine) on coreference between gendered pronouns, while they use gender dictionaries to infer gender of words that are not explicitly gendered. We also require agreement of grammatical gender (neuter/common) which is not a distinction that is present in English.

Antecedent Ordering

When considering candidate antecedents for a mention, the order of consideration matters. The original algorithm uses a syntax parser to get syntax trees for all the clauses. For every clause in the same sentence as the mention they then perform a left-to-right breadth first traversal and take the antecedents in that order. This is intended to favor syntactic salience by favoring noun phrases closer to the top of the tree, and subjects by favoring noun phrases that occur early in the clause [2]. For antecedents in other sentences they order them by textual proximity to the mention.

We only had access to a dependency parser and no syntax parser. Due to this we changed the ordering of mentions of the same sentence to prioritize mentions whose head words have the subject relation. This captures part of the original intent, but not the syntactic salience part.

4.3 Machine Learning Algorithm

The machine learning approach is largely based on the algorithm described in [3]. Just like our rule-based approach, this algorithm uses a sieve based, incremental system. Each mention starts in its own cluster, and these clusters are then gradually merged. The sieves are applied in order, and for each sieve we attempt to find an antecedent for every mention. Here the approach differs however. Rather than using rule-based boolean functions, each sieve consists of a random forest that has been trained to detect coreferent anaphor-antecedent pairs. We consider every candidate antecedent of the anaphor that is within a certain distance, and choose the one that has the highest probability to be coreferent, according to the random forest. To avoid overclustering the probability must meet a certain threshold, which was roughly tuned on the

development set.

Algorithm 3: ML-Based Algorithm

```
Input: document, sieves, predictedMentions (from mention
       prediction or gold mentions)
Output: predicted clusters for document
Put every mention in its own cluster;
for sieve in sieves do
    for mention in predictedMentions do
        if mention.POS ≠ sieve.anaphorPOS then
            continue;
        end
        bestAntecedent = None;
        bestProbability = 0;
        for antecedent in
         GetCandidateAntecedents(sieve.maxSentenceDistance,
         mention, predictedMentions) do
            if antecedent.POS ≠ sieve.antecedentPOS then
                continue;
            end
            p := CoreferenceProbability(sieve, mention, antecedent);
            if p > bestProbability then
                bestAntecedent = antecedent;
                bestProbability = p;
            end
        end
        if bestProbability ≥ sieve.probabilityThreshold then
            MergeClusters(mention, bestAntecedent);
        end
    end
end
```

Each sieve has been trained for, and is used for, a certain combination of POS-tags. The following sieves are used, in the given order (intended to put higher precision sieves first):

- Proper: Proper noun antecedent and proper noun anaphor
- Common: Common noun antecedent and common noun anaphor
- Proper-Common: Proper noun antecedent and common noun anaphor

Algorithm 4: GetCandidateAntecedents

```
Input: maxSentenceDistance, mention, predictedMentions
Output: candidate antecedents for mention
for otherMention in predictedMentions do
    if otherMention comes before mention then
        if mention.sentenceId - otherMention.sentenceId <
         maxSentenceDistance then
            candidateAntecedents.append(otherMention);
        end
    end
end
return candidateAntecedents;
```

Algorithm 5: CoreferenceProbability

```
Input: sieve, mention, antecedent
Output: Probability of coreference according to the sieve's random
        forest model
featureVector := [];
for feature in sieve.selectedFeatures do
    featureValue := calculateFeature(feature, mention, antecedent);
    featureVector.append(featureValue);
end
p := sieve.randomForestModel.getClassProbabilities(featureVector);
return p["coreferent"];
```

- Pronoun: Any kind of antecedent and pronoun anaphor

Each sieve consists of a random forest that has been trained on the training portion of the data, for more details see the Training section. For the implementation we use the scikit-learn random forest [40]. The sieves as implemented in the code also contain information such as probability thresholds required, and which anaphor and antecedent POS they require. This is denoted as sieve.probabilityThreshold, sieve.antecedentPOS etc. in the pseudocode.

4.3.1 Features

The random forests require a feature vector representation of each anaphor-antecedent pair. Some of the features we use are based only on the anaphor or the antecedent, some on the combination of them, and some on their clusters combined or individually.

The features are taken from the algorithm presented in [3] as well as from the set of features for Swedish presented in [38]. For a full list of the features used, see Table 4.1. Features that check if strings are identical or not are always case-insensitive. The features that are marked as boolean also have a third value indicating that the feature is not applicable for the antecedent-anaphor pair.

All features are passed to the model in numerical format. String values are encoded using one-hot encoding, meaning that each possible value gets its own boolean feature. Due to this the total amount of features is very large, making it necessary to use feature selection. Note that even though random forests are good at identifying useful features by themselves, it is necessary to perform this preprocessing step, as the generated feature vectors would otherwise consume too much memory.

Feature Selection

The purpose of the feature selection is to identify a good set of features to use for the feature vectors of each sieve. This is done before the models are trained, as the selected features are both used for training and any subsequent prediction that uses the trained model. Since different sieves may benefit from different features, the selection is done for each sieve independently. For a feature to be selected, it needs to meet a threshold when it comes to pointwise mutual information between the feature and the corresponding class, as calculated on the training data. This means that features that are unlikely to

Table 4.1: List of all features.

Name	Description	Type
sentenceDistance	Sentence distance between the mentions	int
mentionDistance	Mention distance between the two mentions, based on antecedent ordering. This is the same antecedent ordering as in the rule-based algorithm	int
minimumClusterDistance	Minimum sentence distance between any mentions in the two clusters	int
mentionClusterSize	Current number of mentions in the anaphor cluster	int
antecedentClusterSize	Current number of mentions in the antecedent cluster	int
numberMatch	Number agreement between the mentions	bool
naturalGenderMatch	Natural gender agreement between the mentions	bool
genderMatch	Grammatical gender agreement between the mentions	bool
animacyMatch	Animacy agreement between the mentions	bool
nerMatch	NER tag agreement between the mentions	bool
identicalHeadWords	Head words are identical	bool
identicalHeadWordsAndProper	Head words are identical, and both are proper mentions	bool
clusterGenitiveMatch	Clusters match if the head word in one cluster is equal to a head word in the other cluster plus an 's'-suffix	bool
exactStringMatch	The anaphor and antecedent strings are identical	bool
clusterHeadWordMatch	The head word of some mention in the anaphor cluster is identical to the head word of some mention in the antecedent cluster.	bool
clusterProperHeadWordMatch	Like clusterHeadWordMatch but the mentions must be proper nouns	bool
clusterGenitiveHeadWordMatch	The head word of some non-pronoun mention in the anaphor cluster is identical to the head word of some non-pronoun mention in the antecedent cluster, if an 's' is appended to one of the head words	bool
clusterLemmaHeadWordMatch	Head word match between clusters but with lemmatized head words	bool
wordvecHeadWordDistance	The distance between the wordvectors of the anaphor and antecedent head words	float
anaphorFirstInSentence	The anaphor mention is at the beginning of a sentence	bool

Name	Description	Type
antecedentFirstInSentence	The antecedent mention is at the beginning of a sentence	bool
dependentOnSame	The head (in the dependency tree) of the antecedent head word is the same as the head of the anaphor head word	bool
antecedentLength	Number of words in the antecedent	int
anaphorLength	Number of words in the anaphor	int
personMatch	The mentions match in person (i.e. first, second or third person)	bool
iWithinIClusterCheck	Some mention in the antecedent cluster is a part of some mention within the anaphor cluster or vice versa	bool
antecedentCluster IncludesAnaphorCluster	Every word in the anaphor cluster that is longer than 3 characters, also occurs in the antecedent cluster	bool
mentionDeprel	The name of the dependency relation the anaphor head word has to its head	string
antecedentDeprel	The name of the dependency relation the antecedent head word has to its head	string
mentionNextWordPos	Part-of-speech of the word coming immediately after the anaphor mention	string
antecedentNextWordPos	Part-of-speech of the word coming immediately after the antecedent mention	string
antecedentGrammaticalGender	Grammatical gender (neuter/common) of the antecedent	string
anaphorGrammaticalGender	Grammatical gender (neuter/common) of the anaphor	string
anaphorDefiniteness	The definiteness of the anaphor head word	string
antecedentDefiniteness	The definiteness of the antecedent head word	string
anaphorPronounType	The pronoun type of the anaphor, if it is a pronoun	string
antecedentPronounType	The pronoun type of the antecedent, if it is a pronoun	string
anaphorCase	The case of the anaphor	string
antecedentCase	The case of the antecedent	string
antecedentPronounText	The full mention string of the antecedent if it is a pronoun, "UNDEFINED" otherwise	string
anaphorPronounText	The full mention string of the anaphor if it is a pronoun, "UNDEFINED" otherwise	string

say something about whether the pair is coreferent is filtered out. For boolean features there is a second condition that must hold, which is that both the "true" and "false" values must occur a certain amount of times among the training examples. This filters out features that almost always have the same value, for example features generated by one-hot encoding string features for rare strings. For the thresholds used for feature selection, see Section 4.3.3.

4.3.2 Training

The models are trained using a set of examples extracted from the training documents. For each mention, the closest coreferent antecedent is chosen as positive example, and any non-coreferent mentions that come between them are chosen as negative examples. As an example, we can look at this sentence:

> [Alice and Eve]$_x$ spent [the morning] fishing on [the lake] before [they]$_x$ left [the cabin].

In this case "they" has its closest coreferent antecedent in "Alice and Eve". Between them comes two mentions: "the morning" and "the lake". So for the mention "they" the algorithm adds one positive example, "Alice and Eve"-"they", and two negative, "the morning"-"they" and "the lake"-"they". Other mentions in this sentence might also have coreferent antecedents in earlier sentences, in which case examples would be generated from these pairs too.

Each sieve is then trained on the subset of these training examples where the mention pair matches the requirements of the sieve. For example the pairs above would be used to train the pronoun sieve, since the anaphor is "they". Since some sieves will have excessive amounts of training examples, subsampling can be enabled for individual sieves. The subsampling process randomly selects 20 percent of the negative examples, and all of the positive examples. The random forest is then trained with these examples. Then, all of the negative examples are passed to this model, and the 20 percent that are the hardest to predict correctly (i.e. gets the highest probability of being positive) are selected as the new examples. The model is now retrained with all the positive examples, and these difficult negative examples.

The sieves are trained in the order they will be used during prediction, and prediction is performed by each sieve as soon as it is trained. The reason for this is that some of the features used are based on clusters rather than individual mentions, and thus training on clusters that have been built by the earlier sieves mimics the prediction situation.

When training the model for use on predicted mentions, mention prediction is run before training starts. A mention matching is also performed, giving a translation between gold and predicted mentions. When checking if two mentions belong to the same gold cluster, the check will default to false if one or both mentions have no corresponding gold mention.

4.3.3 Parameters

This algorithm has many different parameters that need to be set, and that will affect the performance of the algorithm. This section describes the settings that were used in the final version of the algorithm.

Feature Selection Parameters

There are two feature selection parameters that can be tuned. The first one is the minimal number of occurrences, i.e. how many pairs in the training data has to exhibit the feature. The second is the pointwise mutual information between the feature and the coreferent/not coreferent label. These parameters were set to 30 and 0.0001 respectively, using the parameters from [3].

Random Forest Tuning

The random forests contain 100 trees. This was not finely tuned, but was found to be a reasonable compromise between performance and training time. With more resources it would likely be beneficial to increase the amount of trees. In [3], increasing the number of trees from 100 to 1000 resulted in some improvement. The tree depth was set to be unlimited, just like in the original algorithm. No extra requirements on splitting were used.

Merging Thresholds

The merging thresholds were set to 0.2 for prediction on gold mentions, and 0.15 on predicted mentions. This means that an anaphor cluster will not be merged with a candidate antecedent cluster if the probability of coreference is lower than 20% or 15% respectively, even if all the other candidate antecedents have lower probabilities.

Sentence Distance Limits

Maximum sentence distance between anaphor and antecedent was set to 5 for the pronoun sieve, unlimited for the proper-proper sieve and 15 for the other

sieves.

4.3.4 Differences From the Original Algorithm

In this section we will describe the major differences between the original algorithm, as described in [3], and our implementation.

Sieves

One major difference between our algorithm and the original, is the lack of rule-based sieves. The rule-based sieves used in the original are a precise constructs sieve, and a discourse processing sieve. These sieves are the same as the ones that were not used in the rule-based algorithm, and were removed for the same reasons. The random forest based sieves used are the same as in the original algorithm, except for a sieve that matched list expressions which was not used in our implementation.

Features

As previously mentioned, some new features that we thought would be useful for Swedish coreference were introduced. Some of the original features were also not implemented, for the following reasons:

- The feature required complex procedures to calculate, such as speaker detection or a constituency tree. These were omitted due to time restrictions on this project.
- The feature was not applicable to our dataset. This applied to features such as document sources and whether the document was conversational text.
- The description did not give enough detail to be able to replicate the feature.

4.4 Implementation

The entire system is implemented in Python, using various tools and libraries. This section will describe the system architecture, as well as the libraries and tools.

4.4.1 Tools, Libraries and Resources

Stanza

Stanza is a natural language processing system for Python developed by the Stanford NLP group. It uses multiple neural network components, that together provides a pipeline that splits a text into sentences and words, and annotates it with a dependency graph, POS-tagging, morphological information and lemmatized forms of the words [41].

Named Entity Recognition

Named Entity Recognition is the task of identifying so called named entities in the next. A named entity can be for instance the name of a person, a date or an organization. For named entity recognition we use KB-BERT, a transformer model that has been trained for named entity recognition in Swedish. The model can recognize entities of the following types: person, organization, location, events and times [42].

Animacy Detection

The animacy feature is not given by any of the tools, and has to be derived using some heuristics. If the mention is a pronoun, it is considered animate if the pronoun is listed as animate. Examples of animate pronouns are "jag" ("I"), "du" ("you", singular) and "hans" ("his"). If the mention is a named entity, it is considered animate if the named entity type is person or organization, and inanimate if it is time, event or location. In all other cases the animacy is marked as unknown.

Word Vectors

For the ML algorithm we need word vectors to generate some of the features. The word vectors used in our implementation are word2vec Continuous Skipgram vectors trained on the Swedish CoNLL17 corpus, downloaded from the Nordic Language Processing Laboratory shared repository* [43]. To handle the word vectors and calculate distances between them we use the Gensim library [44].

* http://vectors.nlpl.eu/repository/

Scorer

For evaluation we have used the official scorer from the CoNLL-2011/2012 Shared Task. The scorer accepts gold and predicted clusters in the official CoNLL format. It scores the results using the following metrics: MUC, B^3, CEAF (mention-based and entity-based) and BLANC. The CoNLL score is then calculated as the average of MUC, B^3 and $CEAF_E$ [45].

4.4.2 System Overview

Both the rule-based and the ML-based algorithm use a single prediction pipeline that handles every step from data preprocessing to viewing the predicted clusters and outputting the clusters in a format ready for scoring. See Figure 4.2 for a visual overview.

The pipeline accepts a JSON file in the format that is exported from Textinator. The file contains multiple documents, where each document consists of the raw text and a series of clusters. Each cluster contains a series of mentions, described by their text, and the start and end position in the raw text. From such a file, the system creates internal representations of the documents. The gold mentions and gold clusters are extracted from the input file and stored in the internal document.

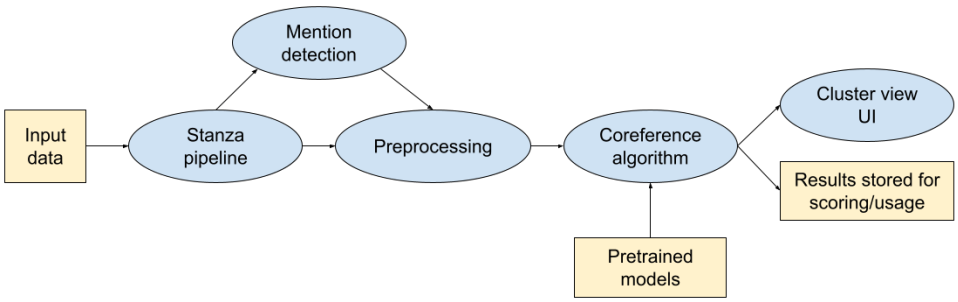

Figure 4.2: Overview of the prediction pipeline. The pretrained models are only used in the case of the ML algorithm. The input data must contain raw text, but may or may not contain gold clusters and/or gold mentions.

The text is now passed through the Stanza pipeline, which annotates the text with tokenization, POS-tagging, morphology analysis and dependency parsing. If the system is not configured to use gold mentions, the mention detection module is run to identify predicted mentions.

The system now extracts some more information about each gold and predicted mention. From the Stanza objects it can extract for instance

lemmatized form of the words, definiteness, number, grammatical gender and head word. We also use KB-BERT to extract NER-tags. All this information will be used by the coreference prediction algorithms; for making decision in the rule-based method, and for generating the feature vectors in the ML method.

The documents are now ready for coreference prediction, and gets passed to either the rule-based or the ML algorithm. When the algorithm is finished, the documents have associated predicted clusters, in addition to the gold clusters.

The final step is to provide evaluation and analysis opportunities. First, we try to match the predicted mentions with the gold mentions. From this matching, the system calculates mention detection recall and precision. Then the gold clusters and predicted clusters are written to a file in the CoNLL-format accepted by the official ConLL-2012 Shared Task Scorer, which can be run to get MUC, B^3, CEAF and BLANC scores. The predicted clusters can also be viewed and compared to gold clusters in a simple user interface, which helps getting an intuition for the quality and the type of errors that have occurred.

Chapter 5

Results and Analysis

In this chapter we will present and analyze the performance of our algorithms, as well as the results of some ablation studies and other experiments that were done during the course of the study.

5.1 Mention Detection

Measuring the precision of the mention detection system is hard. Since we only have gold annotations for mentions that belong to a coreference cluster, any singleton mentions detected will be recorded as precision errors. The goal of the mention detection is primarily to achieve a high recall, since singleton mentions should be filtered out by the coreference resolution algorithm. Recall errors on the other hand, can not be fixed by the coreference algorithms, and will result in coreference recall errors. For the development set we had a mention detection precision of 25% and a recall of 83%.

5.2 Ablation Studies

In order to get some insights into how different parts of the algorithms contributed to the performance, ablation studies were performed. These studies were done on the development set, to mimic the constraints in the CoNLL Shared Task. The best performing settings from each ablation experiment was then run on the test data to give the final results, see 5.3

5.2.1 Rule-based algorithm

All scores are measured on the development set. A base version of the algorithm was chosen and then modified, as described below.

- Base: Uses the exact match sieve, the genitive resolution sieve and the pronoun resolution sieve. Relaxed string match and lemmatized head word match are not included. Subjects are prioritized for close antecedents. This setting is used as the base for the others.
- -exact: No exact match sieve.
- -genetive: No genetive resolution sieve.
- -pronClust: No pronoun resolution sieve.
- -cluster: The cluster based pronouns resolution sieved is replaced by a pronoun resolution sieve that only consideres the mentions, not their clusters.
- -subjectPrio: No subject prioritization for close antecedents. Instead, mentions that appear early in the sentence are prioritized.
- +lemmaHw: The lemmatized head word match sieve is added.
- +relaxed: The relaxed string matching sieve is added.
- +noIndefAna: Disallow matching for indefinite anaphors.

The algorithm was evaluated on both predicted mentions, shown in Table 5.1, and on gold mentions as shown in Table 5.2. When used on gold mentions, the relaxed string match and lemmatized head word match sieves were expected to be beneficial to use, and are as such included in the base version of the algorithm.

Table 5.1: Rule-based algorithm evaluated on predicted mentions.

	MUC			B^3			$CEAF_e$			
System	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**	**CoNLL F1**
Base	43.45	42.73	43.09	34.5	40.76	37.37	45.94	29.82	36.16	38.87
-exact	21.44	35.32	26.68	15.83	37.66	22.29	26.71	24.7	25.67	24.88
-genetive	40.38	41.42	40.9	31.94	40.67	35.78	44.71	28.88	35.09	37.26
-pronClust	29.52	**57.92**	39.11	23.02	**55.06**	32.47	28.95	**38.32**	32.98	34.85
-cluster	43.45	42.73	43.09	34.52	40.55	37.29	45.72	30.26	36.41	38.93
-subjectPrio	43.17	42.46	42.81	34.2	40.43	37.05	45.57	29.44	35.77	38.54
+lemmaHw	**45.4**	37.55	41.1	**36.43**	35.06	35.73	46.09	27.41	34.38	37.07
+relaxed	44.28	39.35	41.67	35.73	37.28	36.49	**47.99**	28.66	35.89	38.02
+noIndefAna	43.45	47.12	**45.21**	34.31	45.47	**39.11**	45.8	32.71	**38.17**	**40.83**

Table 5.2: Rule-based algorithm evaluated on gold mentions.

	MUC			B^3			$CEAF_e$			
System	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**	**CoNLL F1**
Base	65.45	88.01	75.07	54.48	87.37	**67.11**	**63.26**	70.58	66.72	**69.63**
-exact	64.9	87.92	74.67	53.78	87.49	66.61	62.36	70.15	66.03	69.1
-genetive	65.18	87.96	74.88	54.39	87.3	67.03	**63.26**	71.17	**66.98**	**69.63**
-pronClust	43.17	**96.27**	59.61	35.98	**96.47**	52.41	42.68	**74.84**	54.36	55.46
-cluster	**65.73**	88.05	**75.27**	**54.62**	86.99	**67.11**	62.31	70.69	66.24	69.54
-subjectPrio	65.45	88.01	75.07	54.48	87.37	**67.11**	**63.26**	70.58	66.72	**69.63**
-lemmaHw	62.67	87.89	73.17	50.82	87.78	64.37	62.29	69.5	65.7	67.75
-relaxed	64.06	88.12	74.19	52.87	87.39	65.88	61.4	70.84	65.78	68.62
+noIndefAna	65.18	87.96	74.88	54.08	87.3	66.79	62.52	70.33	66.2	69.29

Analysis

One interesting aspect to discuss is the effect of the exact match sieve, combined with the lemmatized head word match and the relaxed string match. On predicted mentions, adding the two latter decreases the F1 score. This is likely due to the over-generation of mentions in the mention detection procedure. Mentions that only match partially are unlikely to be relevant, and it is more effective to focus on exact matches. For gold mentions however, both of these sieves improve the result slightly. If we know that a mention is correct and should be coreferent with something, it is very likely that a partially identical mention is coreferent. Since having gold mentions is unrealistic for real coreference systems, the interesting question is how good mention detection has to get for these sieves to become beneficial.

Another setting that is clearly beneficial for predicted mentions is to not consider indefinite anaphors. This is reasonable, as indefinite noun phrases usually introduce a new referent, rather than refer back to a previously

mentioned one. For gold mentions this setting decreases the CoNLL F1 score, but only by very little.

Regarding cluster based decisions for pronoun resolution, we can see that it improves the result marginally for predicted mentions but the other way around for gold mentions. Using cluster based resolution was one of the key features of the original algorithm, and as such a greater impact was expected. One possible explanation for this difference is that our algorithm lacks for example the apposition detection, that may allow for clustering mentions that are dissimilar. Most of our sieves are based on string similarity, and thus it is likely that the mentions in a cluster will contribute with largely overlapping information.

5.2.2 Machine Learning Algorithm

For the ablation study on the ML algorithm, the base version uses all sieves, and the other versions are without one sieve, as listed in the table. "P/C" is the proper-common sieve. Finally, the last test uses all the sieves but removes the word vector feature. The algorithm was evaluated on both predicted mentions, shown in Table 5.3, and on gold mentions as shown in Table 5.4. The experiments on gold mentions were run with subsampling turned off.

Table 5.3: Machine learning algorithm evaluated on predicted mentions.

	MUC			B^3			$CEAF_e$			
System	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**	**CoNLL F1**
Base	40.94	62.55	49.49	30.6	54.81	39.27	**36.61**	43.74	39.85	42.87
-Proper	33.7	57.34	42.45	24.44	50.87	33.02	31.26	39.44	34.87	36.78
-Common	24.51	64.7	35.55	18.27	59.38	27.94	21.25	45.55	28.98	30.82
-P/C	**41.22**	62.97	**49.83**	**30.7**	54.67	39.32	36.38	44.64	**40.09**	**43.08**
-Pronoun	24.51	**70.96**	36.43	18.36	**67.26**	28.85	23.67	**48.42**	31.8	32.36
-WordVecs	40.11	61.53	48.56	29.19	54.27	37.96	33.82	42.67	37.73	41.42

Table 5.4: Machine learning algorithm evaluated on gold mentions.

	MUC			B^3			$CEAF_e$			
System	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**	**CoNLL F1**
Base	71.58	83.71	77.17	64.36	75.28	69.39	59.19	74.68	66.04	70.87
-Proper	60.44	80.97	69.21	54.22	71.3	61.59	52.22	71.93	60.51	63.77
-Common	35.93	**85.43**	50.58	29.36	**81.4**	43.16	31.26	66.98	42.62	45.45
-P/C	**72.14**	85.19	**78.12**	**64.81**	77.51	**70.59**	**61.2**	**75.8**	**67.72**	**72.14**
-Pronoun	46.23	83.41	59.49	41.93	79.03	54.79	44.4	74.94	55.77	56.68
-WordVecs	70.19	81.02	75.22	63.72	72.84	67.98	56.68	75.76	64.84	69.35

Analysis

An interesting result that can be noted here is that removing the Proper-Common sieve actually improves the total performance slightly. There are multiple possible explanations for this. Primarily, this type of coreference is hard to algorithmically detect, as it often requires real-world knowledge. Unlike pronoun resolution, the coreferent mentions can be very far apart, and unlike common-common and proper-proper resolution the words are often completely different. Another factor that likely contributes is that it is the least common type of coreference, so this sieve has the smallest amount of training data.

We can also note that removing the word vectors decreases the performance, but the difference is still fairly small. This result is of interest since the word vector feature introduces the greatest risk for bias. See Chapter 6 for further discussion of this topic.

5.3 Final Results and Comparison to Other Studies

Table 5.5 shows the scores of the selected best version of the ML and rule-based algorithms respectively (ML/Rule) on gold and predicted mentions (G/P). The ML algorithm is trained on the combined training and development sets, and both algorithms are evaluated on the test set.

The versions used are the base versions as described above, with the following changes. For the ML algorithm, the version without the proper-common sieve was used, since it achieved a better total score in the ablation study on the development set and particularly improved precision, which is often more important than recall in coreference resolution. Subsampling was used when using predicted mentions, but not for gold mentions. For the rule-based algorithm, the indefinite anaphor removal is used.

Table 5.5: Final results.

	MUC			B^3			$CEAF_e$			
System	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**	**CoNLL F1**
P/Rule	48.25	31.96	38.45	41.52	29.14	34.25	47.87	25.18	33.01	35.24
P/ML	47.58	45.04	46.28	38.59	39.28	38.93	41.43	36.2	38.64	41.28
G/Rule	68.16	83.58	75.09	59.71	76.85	67.21	56.79	74.8	64.56	68.95
G/ML	76.04	81.15	78.51	67.74	65.01	66.35	50.05	74.55	59.89	68.25

To compare we can look at to the hybrid method by Nilsson, which achieved the following MUC scores on gold mentions: Precision: 67.56 Recall: 66.52 F1: 67.04 [38]. It should be noted that those gold mentions included singletons, making it a harder task than our gold mention task, where only coreferent gold mentions were given. Additionally, the algorithm was evaluated on a different corpus. It is thus hard to make any comparison, since the difficulty level of their task lies between our predicted and gold tasks.

It is also interesting to compare to the original versions of the algorithms we have adapted, as well as a state-of-the-art algorithm. Table 5.6 shows the reported results on the CoNLL dataset (using predicted mentions) for the original rule-based algorithm, the original ML algorithm and one of the leading algorithms today.

Table 5.6: Scores of the original algorithms and leading neural algorithm.

	MUC			B^3			$CEAF_e$			
System	**R**	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**	**CoNLL F1**
Lee 2013 [2]	65.08	62.41	63.72	50.23	54.08	52.08	54.01	44.27	48.65	54.82
Lee 2017 [3]	68.75	76.4	72.37	55.82	65.94	60.46	51.99	62.5	56.76	63.2
Neural [35]	81.2	73.6	77.2	72.3	61.7	66.6	65.2	60.2	62.6	68.8

No direct comparison can be made, as the definition of coreference, language and dataset used are different, but it is clear that these algorithms perform significantly better. This was expected, as our algorithms are only partial implementations of the original systems. Further, we had a significantly smaller set of training data.

5.4 Effect of More Training Data on the Results

In order to estimate how much additional training data might affect the performance of the ML algorithm, we experimented with the training set size to see how this affected the CoNLL F1. This experiment was done on gold mentions, using the Base version of the ML algorithm, and was evaluated on the development set. Based on the result, as seen in Figure 5.1, it seems likely that adding more training data would improve the performance significantly.

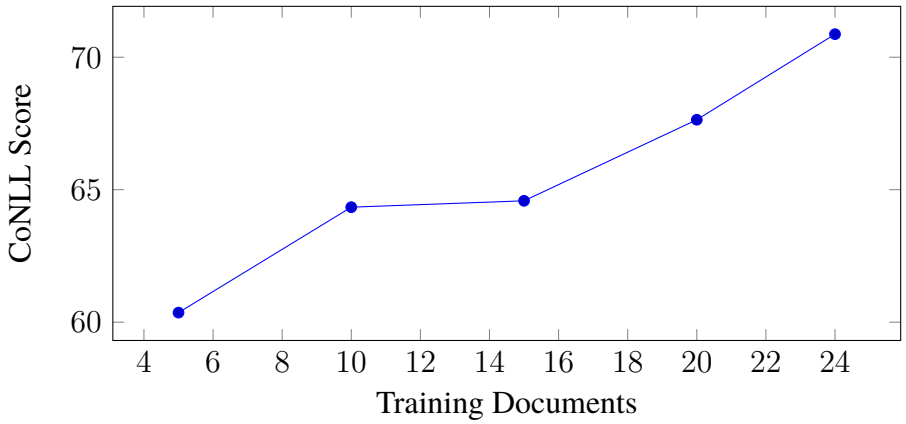

Figure 5.1: Improvement in CoNLL F1 score as the amount of training data is increased. Evaluated on gold mentions on the development set.

5.5 Analysis, Errors and Examples

In this section we will analyze the type of errors the algorithms make, and under what conditions they succeed. Examples from the development set will be given, with English translations. Note that these translations are inexact, and the constructs in Swedish might be different in many ways. For example, in English it might seem like there is another noun between two mentions, when the Swedish version uses a different construction. For this reason, analyzing the English sentences by themselves is likely misleading. All quotes are from Talbanken.

Analysing the ML algorithm used on gold mentions, there are two patterns in the mentions that are marked as coreferent: string similarity/shared head word and the proximity. Precision errors seem to occur often in mentions that are close to each other. For example:

> [År 1970]$_x$ hade ungefär 700000 förvärvsarbete i [vår]$_x$ region.
>
> (In [the year of 1970]$_x$ approximately 700000 people were employed in [our]$_x$ region.)

The first mention is a year, and the second is an animate plural pronoun, so the only reasonable explanation for why they should be coreferent is the proximity.

The preference for similar strings is often effective, since coreference is often repetition. However it sometimes fails, such as linking "landet" ("the country") with "utlandet" ("abroad").

A successful example is this part of a cluster:

> [Soldaterna]$_x$ bör vara tacksamma mot [sin]$_x$ forne kompanichef ... Denne säger till Expressen:
>
> Jag har sett tillräckligt många pansarvärnsinstruktörer stå med handen bakom örat för att kunna höra vad man säger. Det skall inte [mina soldater]$_x$ behöva bli utsatta för.
>
> ([The soldiers]$_x$ should be grateful to [their]$_x$ previous company commander ... He says to Expressen:
>
> I have seen enough instructors standing with their hand behind the ear to hear what you say. That is not something [my soldiers]$_x$ should have to experience.)

However, the cluster unfortunately also contains other mentions of soldiers that are not in fact the same group that is mentioned here, for instance the two mentions in this sentence earlier in the text:

> [En soldat]$_x$ kan bli obotligt hörselskadad redan vid [sin]$_x$ första skjutning.
>
> ([A soldier]$_x$ could get irreversible hearing damage as early as during [their]$_x$ first shooting.)

While their F1 scores are similar on gold mentions, the rule-based version outperforms the ML algorithm in precision (at the cost of recall) in every metric, especially B^3. Why that is the case can be clearly seen when examining the clusters the rule-based algorithm generates, as they are generally smaller and more likely to miss mentions than to have too many.

For example, the mentions in the two previous examples are correctly in separate clusters in this result. However, both of the algorithms also have some extra mentions that should not be included. The rule-based algorithm erroneously includes other mentions of "soldaterna" ("the soldiers"), due to the exact match sieve, that will cluster all mentions that have an exact string match.

For the rule-based algorithm, pronoun resolution seems to dominate precision errors, for example:

> [Skador]$_x$ som [de]$_x$ kommer att få känna av...
>
> ([Injuries]$_x$ that [they]$_x$ will feel...)

Recall errors on the other hand, tend to involve a noun or proper noun anaphor where the mentions are not identical. For example "de tre parterna" ("the three parties") and "de tre" ("the three") were not identified as coreferent by this algorithm, although they should be.

Chapter 6

Discussion

In this chapter we will discuss some of the decisions that were made in this project, and how they affected the results. We will also discuss some of the more interesting results presented in Chapter 5.

6.1 Data Quality and Annotation Issues

The first thing that needs to be discussed is the dataset that was annotated and used, as its qualities and properties will affect the results of the rest of the study. Annotating coreference resolution is not an easy task, as was evident from the inter-annotator agreement statistics from OntoNotes [18]. In our annotation work, we focused on getting sufficient amounts of data with the available resources. This led to some compromises in terms of annotation quality:

- Only one annotator worked on each text.
- Very limited post-checking was done. We only corrected mentions which did not align with word boundaries.
- Limited time was spent perfecting the annotation guidelines.

6.1.1 Error Sources

In discussions between the annotators, some important sources of errors were found. One of them was text length. It was perceived that the time and effort it took to annotate a text did not grow linearly with the length of the text, but much quicker. This is likely due to the fact that coreference can occur between

any parts of the text, meaning that a mention at the end of the text might corefer with one at the very beginning. Having to keep all these mentions in memory, or constantly cross-checking against other parts of the text, is difficult and likely increases the time as well as the amount of errors. How great this effect is was found to vary depending on how many referents was discussed in the text, where texts with more referents are harder to annotate.

Another source of ambiguity was that many of the texts in the corpus vary between talking about a general referent, and a specific one. For example in one text the role of the mother was discussed. In this text "mamman" (the mother) sometimes seemed to refer to all mothers, or the concept of a mother, but at other times referred to a specific mother in a given example. The lines between these cases were often blurry, making it hard to know which coreference cluster a mention should belong to. Similarly, many of the texts were of an instructional type, where it is explained how "you" should fill in a form, submit a sample for analysis etc. References to these things (forms, samples, tax statements, pensions etc.) often seem to concern a specific referent (the one "you" have) but as the explanation explores different options (e.g. pensions for married or unmarried people, positive or negative test results for the sample) it becomes unclear if it can really be the same referent. This type of references are very hard to disambiguate, and are not discussed a lot in previous coreference literature. Likely this is due to them being common to the type of instructional texts and argumentative articles that were common in our dataset, but less prominent in for example fictional texts or news broadcasts, which are common in many other corpora.

Pronouns were in general considered easy to resolve. This was expected, as all the texts are professionally written, and should not have many cases of ambigious pronouns. One issue was with reflexive pronouns, and when they constitute coreference. Overall we decided that they are coreferent when the reflexive pronoun is actually necessary to determine who the action is performed upon, rather than being there as part of a fixed expression. "Han tvättar sig" ("he washes himself" is an example of what we considered coreference. However, "han tänker sig att..." (approximatly "he imagines that...") would not be a case of coreference, as there could be no other pronoun than "sig" (himself) there. It is simply a part of the expression. This distinction will likely be difficult to capture in algorithms.

Determining the appropriate span of a mention was also sometimes difficult. Some noun phrases could have different stretches depending on how many of the descriptions were considered part of the mention. This issue could likely be amended by more specific annotation rules.

6.1.2 Effects on the Results

The annotation quality affects both how accurate the evaluation of the algorithms will be, as well as the performance of the algorithm. If the rules are ambiguous, machine learning algorithms will have difficulty finding patterns in the data, and it will be harder to create good rule-based algorithms.

One example of this is the case of mention boundaries, which were not always clearly defined according to our annotation rules. This meant it was also hard to write the mention detection component in a way that would mirror the decisions of the annotators. This flaw has big impact on the final results, as a difference in span between gold mention and predicted mention means the mention is guaranteed to become either a recall or precision error, no matter how well the coreference algorithm performs.

6.2 Algorithms Discussion

In this section we will discuss some aspects of the algorithms, and their effects on the performance.

6.2.1 Mention Detection

The mention detection component is an important part of the coreference pipeline, that was not given much time or attention in this study. The large difference in performance between gold mentions and predicted mentions indicates that there is much room for improvement. One particular issue is the inability in the system to choose between mentions with the same head word but different spans. For example, in the sentence "She helped the man who had gotten injured", should "the man" or "the man who had gotten injured" be extracted as a mention? In the current system, both might be added. Since the coreference algorithms focus a lot on head word, the algorithm is likely to either cluster both or none of "the man" and "the man who had gotten injured" with a subsequent "he", but one of them will increase the final score of the clustering, and the other decrease it. Part of the solution to this would be, as previously mentioned, to have clearer annotation guidelines and then adapt the rules used for mention detection based on this. Another possible avenue for improvement would be to implement an ML based mention detection component, that uses random forests to choose between different mention spans. This approach was taken in [3].

6.3 Coreference Prediction

As could be seen in the Chapter 5, the ML and rule-based algorithms got similar CoNLL scores when using gold mentions, while the ML algorithm outperformed the rule-based one significantly on predicted mentions. One possible reason for this large difference, is that there is nothing encoded in our rules that is targeted at filtering out incorrect mentions, while the ML algorithm may learn such strategies naturally.

In the small experiment where the ML algorithm was trained on increasing amounts of training data, the CoNLL F1 score showed a steady and promising increase. This indicates that one of the reasons our implementation performed worse than corresponding algorithms for English is likely the much smaller training dataset. It also suggests that given more training data, the ML algorithm could quickly become the better alternative as compared to the rule-based algorithm.

Another important subject to discuss when it comes to comparison of scores between different algorithms, is the tools and existing annotation on the dataset that has been used. In the CoNLL challenges gold annotations on the OntoNotes corpora was used for things such as syntax trees. This ensures that any errors comes from the coreference algorithms themselves, rather than tools earlier in the pipeline. In our case we used a number of tools such as Stanza and KB-BERT, meaning that any errors in them will propagate and potentially cause errors in the coreference prediction.

6.4 Ethics and Sustainability

Ethics is always an important aspect to consider when it comes to natural language processing. Bias is a common problem, and coreference resolution algorithms have previously demonstrated issues when it comes to for example gender bias. The algorithms used in this experiment have not been evaluated for bias, but one benefit is that the algorithms are relatively transparent, and the risk for bias can be analyzed and mitigated. The rule-based version in particular uses explicit rules, of which none could reasonably introduce bias. The ML algorithm is somewhat harder to analyze, but the greatest risk likely lies in the usage of word vectors. As a word vector feature encourages the algorithm to cluster words that the word vectors consider semantically similar, any bias present in the word vectors may propagate to the coreference algorithm. One solution to this could be to use word vectors that have been

debiased or to exclude that feature altogether. Debiasing might be preferential in this case, as removing the word vector feature was shown to decrease the performance significantly.

From a sustainability perspective, a very important subject is that of energy consumption. Machine learning algorithms, and particularly neural networks, often consume a lot of energy during training [46]. The two algorithms presented in this paper both have the benefit of requiring relatively little computational resources. This resource efficiency also gives a benefit of increased equality, as it makes the algorithm easier to adapt for low-resource languages, where high-end computational equipment and large annotated corpora might be hard to finance.

Chapter 7

Conclusions and Future Work

Both the rule-based algorithm and the ML algorithm show great promise for future development of coreference resolution for Swedish. While the ML algorithm primarily needs more training data, improving the rule-based algorithm requires more study and algorithm development.

7.1 Annotation and Data

In the future it would be interesting to see more work in creating a high-quality and reasonably large corpus for coreference resolution. This could either be done by using the lessons learned from this project to create a new, improved set of guidelines and perform a new annotation project. Another option would be to improve on the current corpus. There are a few ways this could be done:

- Going through all the current annotations to check for errors.
- Doing a second annotation pass of all the texts, and then comparing the annotations to catch errors.
- Adding singleton mentions to all the annotations.
- Adding more types of relations, using a different markers. For example copula relations, apposition, part-whole relations.
- Annotating more texts. Based on the experiments on training set size, adding more data might improve the F1 score of the ML algorithm significantly.

7.2 Further Work on the Algorithms

In this study we did some rough tuning of various parameters, and experimented with removing different sieves and features. These changes affected the results significantly, and investing more time in tuning and tweaking the algorithms could likely give valuable performance improvements.

Further, the features used are only a subset of those present in the previous studies, many having been omitted due to time constraints. Some of these omitted features, such as synonymy checks, could provide valuable semantic information that is currently lacking in the algorithms. An interesting future project could be to identify promising new features and implementing them. Implementing new sieves would also be very interesting, and is easily done due to the modular structure of the algorithms.

One significant source of errors are when there are multiple identified mentions that share head word. One way to amend this could be to improve mention detection, for example by implementing machine learning based algorithm. It would also be useful to do a more thorough study of what makes a mention more likely to be coreferent, than another mention with the same head word, and use the findings to improve the rule-based coreference algorithm, or to add more features to the ML based algorithm.

References

[1] K. N. Björkenstam, "SUC-CORE: A balanced corpus annotated with noun phrase coreference," *Northern European Journal of Language Technology*, vol. 3, pp. 19–39, 2013.

[2] H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky, "Deterministic coreference resolution based on entity-centric, precision-ranked rules," *Computational linguistics*, vol. 39, no. 4, pp. 885–916, 2013.

[3] H. Lee, M. Surdeanu, and D. Jurafsky, "A scaffolding approach to coreference resolution integrating statistical and rule-based models," 2017.

[4] R. Sukthanker, S. Poria, E. Cambria, and R. Thirunavukarasu, "Anaphora and coreference resolution: A review," *Information Fusion*, vol. 59, pp. 139–162, 2020.

[5] Cambridge University Press. (2021) "Lemma". [Online]. Available: https://dictionary.cambridge.org/dictionary/english/lemma

[6] U. Teleman, S. Hellberg, and E. Andersson, *Svenska Akademiens grammatik*. Svenska Akademien, 1999. ISBN 91-7227-126-4

[7] K. Fraurud, "Processing noun phrases in natural discourse," Ph.D. dissertation, Stockholm University, 1992.

[8] R. McDonald, J. Nivre, Y. Quirmbach-Brundage, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Täckström *et al.*, "Universal dependency annotation for multilingual parsing," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2013, pp. 92–97.

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[10] L. I. Smith, "A tutorial on principal components analysis," 2002.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[13] T. Bolukbasi, K.-W. Chang, J. Zou, V. Saligrama, and A. Kalai, "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings," *arXiv preprint arXiv:1607.06520*, 2016.

[14] G. Bouma, "Normalized (pointwise) mutual information in collocation extraction," *Proceedings of GSCL*, pp. 31–40, 2009.

[15] R. Grishman and B. Sundheim, "Message Understanding Conference-6: A brief history," in *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996. [Online]. Available: https://aclanthology.org/C96-1079

[16] L. Hirschman and N. Chinchor, "Appendix f: Muc-7 coreference task definition (version 3.0)," in *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*, 1998.

[17] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. M. Strassel, and R. M. Weischedel, "The automatic content extraction (ace) program-tasks, data, and evaluation." in *Lrec*, vol. 2, no. 1. Lisbon, 2004, pp. 837–840.

[18] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes," in *Joint Conference on EMNLP and CoNLL-Shared Task*, 2012, pp. 1–40.

[19] M. Vilain, J. D. Burger, J. Aberdeen, D. Connolly, and L. Hirschman, "A model-theoretic coreference scoring scheme," in *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*, 1995.

[20] A. Bagga and B. Baldwin, "Entity-based cross-document coreferencing using the vector space model," in *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*. Montreal, Quebec, Canada: Association for Computational Linguistics, Aug. 1998. doi: 10.3115/980845.980859 pp. 79–85. [Online]. Available: https://www.aclweb.org/anthology/P98-1012

[21] X. Luo, "On coreference resolution performance metrics," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005, pp. 25–32.

[22] J. R. Hobbs, "Resolving pronoun references," *Lingua*, vol. 44, no. 4, pp. 311–338, 1978.

[23] S. Lappin and H. J. Leass, "An algorithm for pronominal anaphora resolution," *Computational linguistics*, vol. 20, no. 4, pp. 535–561, 1994.

[24] A. Haghighi and D. Klein, "Simple coreference resolution with rich syntactic and semantic features," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 2009, pp. 1152–1161.

[25] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. D. Manning, "A multi-pass sieve for coreference resolution," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010, pp. 492–501.

[26] A. Zeldes and S. Zhang, "When annotation schemes change rules help: A configurable approach to coreference resolution beyond ontonotes," in *Proceedings of the Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2016)*, 2016, pp. 92–101.

[27] W. M. Soon, H. T. Ng, and D. C. Y. Lim, "A machine learning approach to coreference resolution of noun phrases," *Computational Linguistics*, vol. 27, no. 4, pp. 521–544, 2001. doi: 10.1162/089120101753342653. [Online]. Available: https://www.aclweb.org/anthology/J01-4004

[28] J. F. McCarthy and W. G. Lehnert, "Using decision trees for coreference resolution," *arXiv preprint cmp-lg/9505043*, 1995.

[29] C. Nicolae and G. Nicolae, "BESTCUT: A graph algorithm for coreference resolution," in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Sydney, Australia: Association for Computational Linguistics, Jul. 2006, pp. 275–283. [Online]. Available: https://www.aclweb.org/anthology/W06-1633

[30] P. Denis and J. Baldridge, "Specialized models and ranking for coreference resolution," in *Proceedings of the 2008 conference on empirical methods in natural language processing*, 2008, pp. 660–669.

[31] G. Durrett and D. Klein, "Easy victories and uphill battles in coreference resolution," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1971–1982. [Online]. Available: https://www.aclweb.org/anthology/D13-1203

[32] K. Clark and C. D. Manning, "Entity-centric coreference resolution with model stacking," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015. doi: 10.3115/v1/P15-1136 pp. 1405–1415. [Online]. Available: https://www.aclweb.org/anthology/P15-1136

[33] A. Culotta, M. Wick, and A. McCallum, "First-order probabilistic models for coreference resolution," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, 2007, pp. 81–88.

[34] E. R. Fernandes, C. N. dos Santos, and R. L. Milidiú, "Latent trees for coreference resolution," *Computational Linguistics*, vol. 40, no. 4, pp. 801–835, 2014.

[35] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution," *arXiv preprint arXiv:1707.07045*, 2017.

[36] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky, "Stanford's multi-pass sieve coreference resolution system

at the conll-2011 shared task," in *Proceedings of the 15th conference on computational natural language learning: Shared task*. Association for Computational Linguistics, 2011, pp. 28–34.

[37] K. Clark and C. D. Manning, "Entity-centric coreference resolution with model stacking," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1405–1415.

[38] K. Nilsson, "Hybrid methods for coreference resolution in Swedish," Ph.D. dissertation, Department of Linguistics, Stockholm University, 2010.

[39] M. Ariel, "Accessing noun-phrase antecedents," 1990.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[41] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," *arXiv preprint arXiv:2003.07082*, 2020.

[42] M. Malmsten, L. Börjeson, and C. Haffenden, "Playing with Words at the National Library of Sweden – Making a Swedish BERT," 2020.

[43] A. Kutuzov, M. Fares, S. Oepen, and E. Velldal, "Word vectors, reuse, and replicability: Towards a community repository of large-text resources," in *Proceedings of the 58th Conference on Simulation and Modelling*. Linköping University Electronic Press, 2017, pp. 271–276.

[44] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[45] S. Pradhan, X. Luo, M. Recasens, E. Hovy, V. Ng, and M. Strube, "Scoring coreference partitions of predicted mentions: A reference implementation," in *Proceedings of the 52nd Annual Meeting of the*

Association for Computational Linguistics (Volume 2: Short Papers). Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 30–35. [Online]. Available: http://www.aclweb.org/anthology/P14-2006

[46] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," *arXiv preprint arXiv:1906.02243*, 2019.

Appendix A

Annotation Guidelines

A.1 Introduction

The goal of the annotation is to mark "mentions", and then link all mentions that are coreferent, i.e. refer to the same entity. This guide aims to define what types of phrases should be marked as mentions and what kind of relations are considered coreference for the purpose of this annotation. The rules start with the most general and important, and then considers special cases that may be hard or impossible to derive from the general rules.

While singleton mentions are not to be marked, they are often marked up in the examples in this guide, as the examples should be considered small snippets from larger texts. Further, only mentions that are relevant to the point of the example are marked to make the examples less cluttered: this does not mean there might not be other occurrences of coreference in the example.

A.2 Mentions

A.2.1 What is a mention?

- A mention is either a noun phrase (such as "den blåa bilen", "en ung man" or "Peter Svensson") or a possessive (such as "hans", "Peters" or "min"). This excludes verb phrases and longer discourse such as sentences and paragraphs.
- Only mentions that corefer with at least one other mention should be marked.

- Mentions can be nested. In "Peters hus" the entire phrase is a mention, but "Peters" is also a mention on its own.

A.2.2 What should be included in the mention?

- Tight appositions should be included in the mention, and can be identified by the lack of comma. Two examples of tight/restrictive appositions are in "min syster Alice" and "tidningen Illustrerad Vetenskap".
- Preposition phrases are included if they are restrictive modifiers. For example in "den litauiska kommissionen för ekonomisk uppgörelse" it's not enough to include "den litauiska kommissionen" to identity the referent, so the entire phrase should be included in the mention.

A.2.3 Proper names

An exception from the rule about nested mentions are proper names, which are atomic. For instance, in the phrase "Stockholms Universitet" it's not possible to mark "Stockholms" as a mention, since it's just part of a name. If it's unclear whether something is a proper name, assume that it is not.

A.2.4 Temporal expressions

Time expressions can be mentions, including deictic expressions such as "igår" or "nu". Dates are atomic, so "7e november 2020" can be a mention but the "november" that is part of it can not.

A.2.5 Negated expressions

Negated expressions can not be mentions. An example is "inga studenter gjorde läxan". Here "inga studenter" can not be a mention, as it lacks a referent.

A.3 What is coreference?

Coreference is a relation between mentions that have an equivalent referent. This excludes for instance bridging relations and bound anaphora. Coreference is transitive, so the relations form "clusters". In the below example, all mentions marked with x belong to the same coreference cluster, since they all refer to Alice.

> $[\text{Alice}]_x$ var på bio med $[\text{sina}]_x$ vänner, så $[\text{hon}]_x$ kom hem sent.

A.3.1 Anaphora is not always coreference

Some anaphoric relations are coreference relations, but not all. A good test for this is to replace the anaphoric pronoun with the NP, and see if the meaning is retained:

> "Alla föräldrar älskar sina barn" → "Alla föräldrar älskar alla föräldrars barn"

In this case the meaning is not retained, so there is no coreference between "alla föräldrar" and "sina". Compare this to:

> "Sofia älskar sina barn" → "Sofia älskar Sofias barn"

The meaning is retained in this case, indicating that there is corefence between "Sofia" and "sina".

The non-coreference anaphora above is an example of a bound variable pronoun. Another important type of non-coreference anaphoric relations are those that are identity-of-sense rather than identity-of-reference. Example:

> Man har jämfört [den ekonomiska utvecklingen i Sverige]$_x$, med [den]$_y$ i Finland.

A.3.2 Copula constructions

When a copula verb is used, the subject and predicative are not coreferent with each other. References to the subject should be connected to the subject mention, and not the predicative.

In this example "Hon" is the subject, "heter" the copula verb and "Ingrid" the predicative:

> [Hon]$_x$ heter Ingrid. [Hon]$_x$ jobbar som läkare.

The predicative can also be part of coreference chains:

> Hon heter [Ingrid]$_y$. [Det]$_y$ är ett vanligt namn.

Other examples of copula verbs are "vara", "kallas", "bli".

A.3.3 Apposition

Apposition should not be marked as coreference.

> [Sveriges statsminister]$_x$, Stefan Löfven, meddelar att [han]$_x$ kommer sammankalla ett krismöte.

A.3.4 Generic and abstract mentions

Generic mentions are references to classes, rather than sets or singular entities. Generic mentions can corefer with mentions such as definite NPs and pronouns. However, they can not corefer with other generic mentions. Example:

> Museet hade nyligen fått in [nya vaser]$_x$ till sin samling. [Vaserna]$_x$ kom huvudsakligen från Kina. Museichefen kommenterar: 'Vi är mycket glada över att få in [nya vaser]$_y$, [de]$_y$ är ett välkommet tillskott till samlingen'.

In most cases, indefinite plurals as well as indefinite singulars are considered generic.

References to abstract/uncountable things, such as "städning", "konst" and "simkunnighet" are considered generic when they are in their indefinite form.

A.3.5 Underspecified mentions

Sometimes references to sets or individual entities, while not generic, are too undefined or fuzzy to determine exactly which group it refers to. These mentions are underspecified. Example:

> [De äldre]$_x$ förtjänar inte att få frysrätter istället för varm mat.... [De äldre]$_y$ var med och byggde upp vårt land, så [de]$_y$ ska behandlas väl.

In this case, "de äldre" in the first mention could refer to both old people in general, anywhere in the world, and the specific group of people at risk of getting frozen meals. In the second mention "de äldre" could refer to the group getting frozen meals, or the senior population of Sweden. Since it's unclear whether the two groups are identical, they should form separate coreference chains.

When in doubt, aim for precision over recall and don't combine coreference chains of underspecified mentions.

A.3.6 Plural mentions and their parts

Plural mentions corefer with their parts if they occur together, but not if they occur separately. Example:

> [[Anna]$_x$ och Lisa]$_y$ åkte på semester. [Systrarna]$_y$ hade kul, men [Anna]$_x$ åkte ändå hem tidigt.

A.3.7 Metonomy

Metonomy is coreference. In the example below there is metonymic coreference between the first "Moskva" and "Ryssland", as they refer to the same entity. The second "Moskva" however, is not a metonymic reference to Russia, but refers to the actual city.

> [Moskva]$_x$ meddelade under gårdagen att de ämnar att ingå i avtalet. Därmed blir [Ryssland]$_x$ det tredje landet som skriver under. Ett första möte kommer ske i [Moskva]$_y$ i september, för att diskutera detaljerna.

A.3.8 Deictic pronouns

Deictic pronouns, such as "vi" in a scientific paper, or "jag" in a novel, should be marked as coreferent to each other. If there is a clearly identifiable organization or person this refers to, other mentions of this is also coreferent to the deictic pronouns.

A.3.9 "Man" as a pronoun

"Man" as a pronoun is normally not coreferent with anything. An example is in "Man blir så trött på det här vädret". This pronoun can not corefer with anything, as it lacks a referent. There are however cases where there is a clear referent, and in such cases there can be coreference:

> [Företaget]$_x$ uttalade sig på tisdagskvällen om skandalen. [Man]$_x$ hävdade att…

A.3.10 Quantified expressions

Quantified expressions can be coreferent. I "Några av forskarna" both the entire expression and only "forskarna" can be mentions. They are however not coreferent with each other. Example:

> [Några av [forskarna]$_x$]$_y$ åkte på konferens, och [några av [forskarna]$_x$]$_z$ jobbade i labbet.

If a quantified expression indicates the entire group, such as in "alla forskarna", the entire phrase should be marked as a mention to be used for coreference.

A.3.11 Function type expressions

Function type expressions can be marked as mentions, and are coreferent with each other if they refer to the same expression.

> "[Antalet röster]$_x$ var 2700. Förra året var [antalet]$_x$ 3000".

A.3.12 Proper nouns in indefinite forms

When proper nouns are used in indefinite form they don't corefer with the normal usage of the proper nouns. So in this case the marked mentions do not corefer:

> "[Stockholms]$_x$ vägar är till stor del utformade för biltrafik. Vi vill ha [ett Stockholm]$_y$ där cyklarna har en självklar plats".

TRITA – EECS-EX-2022:52