Singleton detection using semantic word vectors and neural networks

Hessel Haagsma

Master's Thesis
Human-Machine Communication

Summer 2015

singleton, noun:
1. A single person or thing of the kind under consideration
2. A child or animal born singly, rather than one of a multiple birth
3. (informal) A person who is not married or in a long-term relationship
4. (in card games) A card that is the only one of its suit in a hand
5. (mathematics) A set which contains exactly one element
6. (linguistics) A noun phrase or other mention that is not coreferential with another noun phrase or mention

Supervisor: Jennifer Spenader, PhD
Department of Artificial Intelligence
University of Groningen

university of groningen

faculty of mathematics and natural sciences

Contents

Abstract

Using semantic word vectors and neural networks, a state-of-the-art singleton detection system is developed. Singleton detection is a pre-processing task for coreference resolution, which aims to filter out singleton mentions (mentions not involved in a coreference cluster). Filtering out singleton mentions reduces the search space of the coreference resolution system, which helps to improve performance.

Instead of using features present in previous approaches (part-of-speech tags, surface semantics, named entity information, etc.), we make use of semantic word vectors, a relatively new technology which represents words as real-valued, high-dimensional dense vectors. These vectors have shown promise as features in other NLP tasks, such as named-entity recognition, paraphrase detection, syntactic parsing and sentiment analysis. In addition to word representations, a recursive autoencoder is used to generate vector representations for mentions consisting of multiple words. These features are used in a multi-layer perceptron classifier to achieve state-of-the-art performance (79.5% overall accuracy) in singleton detection on the CoNLL 2011 and 2012 Shared Task data (i.e., the OntoNotes corpus).

It is shown that off-the-shelf semantic word vectors are information-rich features, which can be used for more than just 'low-level' syntactic NLP tasks, but also for tasks more semantic in nature, such as singleton detection. This shows their promise for further advances in natural language processing.

It is hypothesized, because semantic word vectors contain more semantic information than other commonly used features, and because they are types of features not already used by coreference resolution systems, that a singleton detection system based on this can benefit coreference resolution more than earlier mention filtering systems. To this goal, performance is evaluated with the most recent versions of the Stanford and Berkeley coreference resolution systems, which are among the state-of-the-art in English coreference resolution. Performance with the Stanford system is good (0.7 point increase in CoNLL F1-score), but not with the Berkeley system (0.3 point increase). As such, the conclusion has to be drawn that, as they are used in this study, semantic word vectors do not have significant added value for coreference resolution performance.

1 Background

1.1 Introduction

Coreference resolution, the identification and linking of all expressions in language that refer to the same entity, is an essential part of language understanding. As such, high-quality automatic coreference resolution is an important task within the field of natural language processing. Natural language processing, or NLP, is the field of science that aims to automatically and computationally understand natural language. Consider Example 1 (matching indices indicate coreferential expressions), for example. In order to make sense of this scene one would have to understand that *he* refers to *Bob* rather than to *Mary*.

(1) [Bob]$_1$ knowingly looked at [Mary]$_2$ on the other side of the room. Then, it seemed like [he]$_1$ winked.

In this case, this is quite an easy task: figuring out that *he* is masculine, so is *Bob*, but that *Mary* is not, should do the trick. However, it is not always that simple. Example 2 is only a slight variation on Example 1, but here, it is much harder to know who *he* refers to.

(2) [Bob]$_1$ knowingly looked at [John]$_2$ on the other side of the room. Then, it seemed like [he]$_1$ winked.

Coreference resolution is essential to almost any high-level natural language processing task. In machine translation, for example, knowing what a pronoun refers to is necessary to be able to produce the correct pronoun in the translated text. Similarly, for automated question answering, coreference resolution is important to both interpreting the question and extracting the knowledge necessary to produce an answer. If a user asks the question in Example 3, for example, the system has to know that *it* refers to *My internet* before it can find an answer.

(3) [My internet]$_1$ isn't working properly. How can I fix [it]$_1$ without having to reset my modem?

The task of coreference resolution essentially consists of two parts: the finding of all referring expressions (called 'mentions') in a text and the clustering of those mentions that co-refer (refer to the same entity). So, for Sentence 2, the mentions are *Bob*, *John* and *he*, and *Bob* and *he* would have to be clustered together, and *John* put in a separate cluster. Although most work in coreference resolution focuses on the clustering part, the finding and filtering of mentions is also important to achieve good performance.

One way in which mention detection and filtering is important is as follows: if words or phrases that do not actually refer to an entity are included in the input for the clustering algorithm, the algorithm might accidentally cluster these expressions with other ones that are referential. For example, *it* in Sentence 2 might then be clustered erroneously with *John*. This results in clusters that have mentions that should not be there, and might also disturb the formation of other correct clusters.

Likewise, performance will be hurt if referential expressions that should have been included are mistakenly left out from the clustering input. For example, if *Bob* is not identified as a mention, *he* might be linked to *John* instead.

In one recent approach to coreference resolution, by Lee et al. (2013), the authors estimate that the inclusion of non-referential mentions in the input for the clustering stage of the algorithm directly causes 14.8% of the error of their coreference resolution system. This is testament to the potential of mention detection and filtering for improvements in coreference resolution performance, and the authors acknowledge this as well, as they state: "*The large number of these errors suggests the need to add more sophisticated anaphoricity detection to our system.*"

Looking at coreference resolution from a larger perspective, it is an interesting task because humans can do it quickly, almost perfectly, and without effort, while simultaneously it is a very difficult task for computers. One reason for this, of course, is that humans have superior language abilities when compared to even the most advanced NLP systems, which helps them makes optimal use of linguistic cues. These cues can be used to resolve Example 1, for example.

More important for coreference resolution, however, is that humans possess a lot of non-linguistic intelligence and knowledge. We have a lot of knowledge of the world and expectations of what is going to be expressed in a text. In Example 4, it helps a lot to know that bank robbers tend to run away after a robbery, and bank clerks usually do not. Computers, on the other hand, do either not have this knowledge or cannot utilize it in an effective way. Usually, they rely on linguistic cues only, basing their predictions mostly on morpho-syntactic and lexical information.

(4) [The bank robber]$_1$ grabbed the money from the clerk's hand. Then, [he]$_1$ ran away quickly, while the alarms sounded.

For mention detection and filtering, the same differences between computers and humans apply, but there is a small difference. The difference is that mention detection is not a task that can be done 100% accurately on its own, i.e. without performing coreference resolution. Even we cannot predict whether a mention is going to be referred to later in a text in all cases. Still, humans are better at mention detection than computers, having no problems identifying, for example, pleonastic *it* in Example 1. The reasons for this are our superior language capabilities, general intelligence and world knowledge.

So, if we want to improve coreference resolution and mention filtering performance, one option is to look at how humans do it and emulate that approach. Obviously, building a complete human-level artificial intelligence would be a good solution, but this is still far in the future. A more feasible idea is to use current natural language processing and machine learning methods, and strive to somehow incorporate world knowledge in these approaches.

One way of doing this is to use lexicalized features, which attempt to do something of the sort. By conditioning features on lexical items (i.e. specific words or word-stems), we can capture word-specific information, which is a form of world knowledge. In Example 4, the existence of *The bank robber* in the sentence could make *the clerk* less likely to be coreferential, conveying the world knowledge that bank robbers are more likely to be the subject of a story than bank clerks. Given enough training examples, a machine learner might be able to acquire this information, but this would require an amount of annotated data that is not currently available.

Therefore, it is preferable to already have this knowledge in some form, and then apply it to the task at hand (such as mention filtering), rather than having to acquire this knowledge during task-specific training. This is, in principle, what humans do, since we already have a lot of general purpose knowledge and can use it to carry out a specific task.

In the case of mention detection, this is expressed in knowing what a text is about, and what is central and what is peripheral to the discourse. When reading Example 4, a human will immediately notice that *the bank robber* and *the clerk* are central to the story, and thus likely to be coreferential mentions. Conversely, it is also clear that *the clerk's hand* and *the alarms* are more peripheral, and thus more likely to be singleton, since they are unlikely to re-occur later in the text. That this works even in a context-less example sentence is testament to the strength of these intuitions.

The approach used in this project aims to do this, exploiting a set of general knowledge for the purposes of a specific task. This is the reason for using pre-trained semantic word vectors (cf. Section 1.4), which capture a large amount of lexical semantic info. This does not amount to the more complex notion of world knowledge, but is the best approximation currently available which is directly applicable to natural language processing.

In this thesis, the hypothesis will be examined that the limitations on current mention detection systems can be overcome by making use of neural networks and semantic word vectors, and can be used to build a state-of-the-art mention detection system. The combination of semantic word vectors and neural networks is up a relatively new machine learning technique, which has successfully been applied to other NLP tasks.

The details of this technique are presented in Section 1.4. First, we will look into mention detection and filtering, and the tasks related to it. In order to have a solid basis for understanding the tasks, and to identify the problems that earlier mention filtering systems have run into, an in-depth study of previous work in the field is presented in Sections 1.2 and 1.3.

1.2 Mention filtering tasks, coreference and anaphoricity

Before we can look at the various mention filtering tasks, there should be some reflection on the definition and usage of the terms anaphoricity and coreference. These are similar but distinct concepts, and defining them clearly helps defining the corresponding tasks better.

An insightful work on this topic is van Deemter and Kibble (2000). Van Deemter and Kibble provide a criticism of the task definition used for the seventh Message Understanding Conference (MUC-7) (Hirschman & Chinchor, 1997). They find that coreference and anaphoric relations are often conflated, causing a range of problems.

They define coreference as follows: *"α_1 and α_2 corefer if and only if Referent(α_1) = Referent)(α_2)"*, where *"Referent(α) is short for the entity referred to by α"*. Anaphoricity is defined, somewhat simplified, as follows: *"an NP α_1 is said to take an NP α_2 as its anaphoric antecedent if and only if α_1 depends on α_2 for its interpretation"*, a definition that leaves some room for ambiguity, since 'depending' can be taken to mean different things.

Keeping these definitions in mind, it is clear that anaphoric and coreference relations are not equivalent. They often co-occur, but there are cases where only one of the two relations holds.

For example, coreferential mentions do not necessarily have to be anaphoric. This is the case with multiple mentions of a named entity, as illustrated in Example 5. Here, the two mentions of *The White House* both refer to the same entity, but are not dependent on each other for their interpretation, while *It* is dependent on other mentions for its interpretation.

(5) [The White House]$_1$ is in Washington D.C. [It]$_1$ is the home of the U.S. president. [The White House]$_1$ was built between 1792 and 1800.

Similarly, anaphoric mentions do not necessarily have to be coreferential. An example of this are so-called bridging anaphora, which are dependent on a previous mention for their interpretation, but do not refer to the same entity. In Example 6, *the frame* is a bridging anaphor, as it is only interpretable because *the bike* is mentioned earlier, even though both mentions refer to different things.

(6) I sold [my bike]$_1$ to my neighbour, because [the frame]$_2$ broke in half yesterday.

Van Deemter and Kibble also provide interesting observations on the treatment of other problematic cases. One problematic case is that of non-referential NPs that are part of anaphoric relations. For example, *a solution* in Example 7 does not refer to a specific thing in the real world, but *it* depends on it for its interpretation.

(7) Whenever [a solution]$_1$ emerged during the meeting, we embraced [it]$_1$.

Another problematic case is that of bound anaphora, as in Sentence 8. Here, it is unclear whether *Every TV network* and *its* corefer, since the first refers to the whole set of TV networks, while the latter refers to only 1 TV network. Similar referentiality problems occur in predicate relations, as with *Higgins* and *the president of Dreamy Detergents* in Sentence 9.

(8) [Every TV network]$_?$ reported [its]$_?$ profits in the year-end report.

(9) [Higgins]$_?$ was [the president of Dreamy Detergents]$_?$ during their glory years.

There is no best way to deal with these problematic cases. Therefore, they are usually dealt with by referring to the guidelines of the corpus that is used for coreference resolution evaluation. This is the approach taken here, too. For this project, the OntoNotes corpus (Weischedel et al., 2013) will be used, so its definitions of coreference (cf. Section 2.1.1 for details) will be used, too.

A final point raised by Van Deemter and Kibble regards what they call markables (mentions). They provide a valuable insight when they state: *"coreference (step 2) helps to determine what the markables are (step 1)"*. This means that the two phases of a coreference resolution system are far from independent. This idea has been brought into practice in coreference resolution systems that attempt to do the two steps simultaneously to improve performance (e.g. Denis and Baldridge, 2007, Cai, Mújdricza-Maydt and Strube, 2011). This is a useful notion to keep in mind when discussing and evaluating the results and limitations of mention detection and filtering systems: mention detection, ultimately, is dependent on coreference resolution.

Using the definition given by Van Deemter and Kibble, we can identify the four different tasks that have to do with the filtering out of mentions. These are all tasks for which systems have been built in previous works, and each concern a different aspect of mention filtering. One such task is anaphoricity detection, which is the task of identifying whether a given mention depends on a previous mention for its interpretation. Another task is antecedent detection, which is the task of identifying whether a given mention is coreferential with a mention later in the same text.

The third mention detection task variant is called non-referential mention detection. This regards the identification of those mentions that do not refer to an entity, e.g. *it* in *'it snows'*. As such, it overlaps with the other two tasks: non-referential mentions are neither anaphoric or coreferential. Nevertheless, it is a conceptually different task, and perhaps slightly easier to solve than the other two tasks. A variant of non-referential mention detection is non-referential *it* detection, which focuses only on non-referential instances of the pronoun *it*, instead of non-referential mentions in general.

A final task to be considered is one that avoids dealing with the notions of coreference and anaphoricity altogether. It covers both anaphoricity and coreference, and is referred to as 'modeling the lifespan of discourse entities' (Recasens, de Marneffe & Potts, 2013), or singleton detection (de Marneffe, Recasens & Potts, 2015). It is defined as a model that is "*[..] not restricted to pronouns or to indefinite NPs, but tries to identify any kind of non-referential NP as well as any referential NP whose referent is mentioned only once (i.e. singleton).*". As such, it overlaps with anaphoricity, antecedent and non-referential *it* detection.

Note that almost all literature on mention filtering deals only with noun-based mentions, and in this thesis, NP-mentions are the only kind of mentions under consideration. There are non-NP mentions though, for example when dealing with event coreference. Here, the mentions do not refer to an entity, but to an event, as in Example 10, where the whole first sentence is the antecedent of *This*.

(10) [The protests in Groningen are escalating]$_1$. [This]$_1$ is problematic for the local police force.

1.3 Previous work

Different variations and specifications of the mention detection tasks have been attempted, using an equally varied number of algorithms and methods. Here, an overview of previous work will be given and insights useful for the current project will be highlighted. An overview and summary of each work discussed here is provided in Table 1 for the reader's convenience.

1.3.1 Detection of non-referential *it*

The earliest work on mention filtering concerns the detection of non-referential *it*, which starts with Paice and Husk (1987). Since non-referential *it* is the most common non-referential mention and it mostly occurs in fixed patterns or expressions, the non-referential *it* detection task established itself as a separate task.

Paice and Husk use a classic, linguistically motivated, rule-based approach. They inspect occurrences of *it* in a corpus, deduct certain patterns from that, and turn these into rules for the detection system. Although this is a very different approach to the one taken here, their analysis can provide useful insights into the task. They note that non-referential *it* is often characterized by having a type of delimiter to its right, such as *that*, *to*, or *whether* and a fixed construction that goes between *it* and the delimiter. For example, a typical usage of non-referential *it* as in Example 11 is formalized as a pattern of the form "it VERB STATUS that STATEMENT".

(11) It is inevitable that I repeat some of the arguments for my hypothesis.

Many more patterns are defined and implementing these yields relatively good performance. Paice and Husk report an accuracy of 86.5% on a corpus of technical documents. The high-precision rules defined by Paice and Husk are useful even for non-rule-based systems, since they provide a good overview of the contexts in which non-referential *it* occurs.

Additionally, we see that their patterns rely on generalization such as 'STATUS' or 'COGNITIVE VERB'. These generalizations are difficult to capture in a feature, and such a feature often relies on a list of words that fall in, for example, the category of cognitive verbs. Since semantic word vectors are strongly suited to capturing similarities between words and grouping similar words together, they are potentially a very suitable way of operationalising these generalizations.

In the context of this project, it is perhaps more interesting to look at learning-based approaches, and see how well they can tackle mention filtering approaches. One such approach to the automatic detection of non-referential *it* was taken by Evans (2001). On the basis of grammars and corpus data, he distinguishes seven types of *it*, two of which, 'pleonastic *it*' (e.g. Example 12) and idiomatic/stereotypic *it* (e.g. Example 13), together form the category of non-referential *it*. The idiomatic/stereotypic category poses a challenge for automatic detection, since the meaning of idiomatic expressions is not always directly based on the meaning of its parts, making them hard to acquire automatically if they do not occur often enough in the data.

(12) It is raining in Baltimore.

(13) I take it you're going home now.

Evans uses the TiMBL memory-based learner (Daelemans, Zavrel, van der Sloot & van den Bosch, 1998). The learner is trained on a large set of features, containing word-positional, pattern-like, lexical, and part-of-speech (POS) type features. Performance on separating referential from non-referential *it* is good, but not directly comparable to that of Paice and Husk (1987), nor to the performance of later systems due to the use of different training and testing corpora. Evans reports an accuracy of 71% for referential/non-referential classification. In 7-way classification, he reports 73% precision and 69% recall for pleonastic *it* but only 33% precision and 1% recall for idiomatic *it*. He used the SUSANNE and BNC corpora, which cover many different genres, e.g. magazine text, newswire and fiction.

A similar approach was taken by Litrán, Satou and Torisawa (2004), who use a memory-based learning (MBL) and a support vector machine (SVM) approach, comparing them directly. The textual domain they operate on is different from other research; a corpus of biological and medical scientific abstracts. Interestingly, they report a high number for the proportion of non-referential usages of *it*: 44%, whereas Evans reported a figure of 29% for his corpus. This might be due to the nature of the corpus used by Litrán et al., it seems likely that scientific abstracts use non-referential *it* more often than most other text types. The set of features (or attributes) used by Litrán et al. is similar to that used by Evans (2001).

Their system shows high classification accuracy, at almost 93%, with the SVM slightly out-performing the memory-based learner. Unfortunately, they report no recall or precision. Performance is remarkably high, but this might be due to specifics of the dataset, and it remains to be seen how it generalizes to other text domains.

Interestingly, Litrán et al. report the most relevant features for both their methods, which provides an extra insight in what is informative regarding the referentiality of *it*. Among the top features are the distance to and the number of following complementizers, the presence of a complementizer followed by a NP sequence and the lemmas of the previous and next verb.

In a more general sense, they conclude that both syntactic and lexico-semantic features are important for the classification of *it*. The same conclusion can be drawn from the work of Evans, which seems to confirm the type of features that can be used to achieve reasonable performance on this task. On the other hand, this also indicates the limit on performance that can be achieved using these features, and the need for different features to push mention filtering even further. Semantic word vectors can capture the information used by Evans and Litrán et al., but contains an additional layer of lexical semantics, thus fitting the requirements for new, better features.

An approach similar to that of Evans was taken by Boyd, Gegg-Harrison and Byron (2005), who also make a multi-type classification of *it*, and like Evans and Litrán et al. use the TiMBL memory-based learner, in addition to a decision tree classifier. Boyd et al. use a more balanced corpus, a subset of the BNC Sampler Corpus (Burnard, 2005), which has the benefit of an extensive set of POS-tags. They report good performance for the TiMBL system: 88% accuracy, 82% precision and 71% recall.

Instead of the seven-way classification of *it* proposed by Evans (2001), Boyd et al. consider only non-referential instances of *it* and distinguish four types, based on an English grammar (Huddleston & Pullum, 2002). These four types are: extrapositional (Ex. 14), cleft (Ex. 15), weather/condition/time/place (Ex. 16), and idiomatic (Ex. 17). Here, too, it can be seen that non-referential *it* can be characterized by syntactic (extrapositional, cleft) and by lexico-semantic means (weather/condition/time/place, idiomatic). Both these types of information are captured by semantic word vectors, making them a promising information source.

(14) It has been confirmed that ...

(15) It is me, Mario!

(16) It snows. It is 8 o'clock.

(17) It was my turn to choose a car.

Although it is certainly useful to look at older work, the most can be gained from looking at the state-of-the-art. By looking at what yields the best performance (so far), we can assess both what is necessary to improve upon that performance and what has been used to achieve state-of-the-art performance. The state-of-the-art in non-referential *it* detection is made up by two papers: Bergsma, Lin and Goebel (2008) and Bergsma and Yarowsky (2011), with the latter being a straightforward extension of, and improvement on, the first. Contrary to previous approaches, Bergsma and Yarowsky rely less on hand-crafted features and more on information extracted from a large corpus. They use a logistic regression classifier, with two kinds of features: N-gram features and lexical features.

The N-gram features are taken from the Google N-gram corpus (Brants & Franz, 2006) and give, for each word, the other most frequent words in it context. Bergsma and Yarowsky exploit this by considering the context in which *it* occurs, checking which other words occur in the same context, and using that information to classify *it* as referential or not. Five types of words that can occur in the same context are identified: *it*/*its*, third-person plural pronouns, other pronouns, unknown words and known non-pronouns. The underlying intuition here is that non-referential *it* tends to occur in contexts that mostly have *it* or other pronouns, while referential *it* occurs in contexts which have more other words.

The lexical features, on the other hand, are more similar to other work, in that they rely on specific words in the mention's context. Bergsma and Yarowsky add all tokens from a large window around *it* as features, and determine the value of these features using supervised learning. Similar to the Boyd et al. (2005) system, they achieve an accuracy of 86%, precision of 82% and recall of 63%. A direct comparison is impossible, since a different corpus (the BBN corpus, Weischedel and Brunstein, 2005) is used.

Nevertheless, it is noteworthy that a generalized, surface-feature-based approach can achieve similar performance to more specific approaches based on linguistically motivated features. Relating this to the current work, a neural network and semantic word vector based system falls squarely in the first category, which makes Bergsma and Yarowsky's results more encouraging.

In conclusion, non-referential *it* detection seems to be a task that is relatively easy to do reasonably well. Using systems that differ in both features and approach to learning/filtering, accuracy scores in the 80% range can be achieved. However, since different corpora and definitions are used by each author, a direct comparison of results is impossible. This also makes the absolute quality of detection systems hard to assess, as not all corpora are balanced across text types and show large variation in the proportion of non-referential *it*'s. Although an accuracy of over 80% seems reasonably high, it does not make for a high-precision, high-recall non-referential *it* filter. For a non-referential *it* detection system to be of real help to coreference resolution, it needs to be better, and this seems to be a lot harder to achieve.

1.3.2 Other mention filtering systems

In addition to the systems focussing on the detection of non-referential *it*, there is previous research on a range of other things that fall under the header of 'mention filtering': the detection of non-referential, non-anaphoric, discourse-new, uniquely identifiable, and non-antecedent mentions, applying it to indefinite NPs, definite NPs or a combination of both.

First, the work of Byron and Gegg-Harrison (2004) should be discussed, since it is closely related to the work on non-referential *it* detection. It focuses on the filtering of non-referential indefinite NPs, and can thus show whether there is a large difference between what works for non-referential *it* and indefinite NPs. They base their approach on the discourse-theoretical work of Karttunen (1976), detecting non-referentiality by considering determiners, predication, negation, apposition, modality, modification and numerals as features.

Instead of training a classifier, Byron and Gegg-Harrison implement a hard filter on some patterns. Hard filtering means simply marking all negated indefinite NPs as non-referential, for example. Their filter removes 12% of all NPs from the resolution input, but this yields only a non-significant improvement in the resolution system's performance. The reason for this lack of improvement is that most of these NPs were already classified as singleton by the coreference resolution system in the first place. Therefore, filtering them out does not improve system performance, although it makes it slightly faster. This is taken as a warning for the current project, namely that there is no one-to-one relation between good mention filtering performance and a significant improvement for coreference resolution performance (cf. also Section 1.3.3).

A different strain of research does not focus on non-referential, but on non-anaphoric and discourse-new mentions. Discourse-new detection is very similar to anaphoricity detection, aiming to identify mentions that introduce a new entity in the discourse. As such, it has a large overlap with anaphoricity detection, and the difference is irrelevant for the most part.

There are two main reasons to look at these different mention detection and filtering tasks. The first is that it shines a light on which tasks have been most successful when it comes to boosting coreference resolution performance. Second, and more importantly, it indicates which kind of approaches and features are relevant for which tasks, and whether there are large differences between e.g. anaphoricity and non-referential *it* detection.

An early work on anaphoricity detection, by Bean and Riloff (1999), focuses on identifying non-anaphoric definite NPs. They use a set of handwritten syntactic heuristics, the assumption that entities in the first sentence of a text are always non-anaphoric, and, interestingly, a measure of how often a given NP is used with and without a definite determiner in a corpus. On a small corpus consisting of short newswire texts, their system achieves 82% precision and 82% recall in classifying definite NPs as non-anaphoric.

In a similar vein, Uryupina (2003) proposes a filtering system for non-anaphoric NPs, both definite and indefinite. In terms of features, Uryupina's system uses syntactic and part-of-speech features, contextual head-matching features and the definiteness probability feature from Bean and Riloff, using the web to estimate probabilities instead of a corpus. Using the Ripper rule-induction system (Cohen, 1995), she achieves 89% precision and 84% recall for discourse-new detection on a subset of the MUC-7 corpus.

Although we cannot compare the performance of Uryupina's and Bean and Riloff's systems directly, we can see that both types of approaches can be successful. The work by Bean and Riloff is very much pattern-based and linguistically informed, similar to that by Paice and Husk (1987), while Uryupina's approach is closer to that by Evans (2001) and Boyd et al. (2005).

Notably, performance on discourse-new detection is similar to that on non-referential *it* detection, and similar approaches and features are used, which we take to indicate that the tasks are not all that different. Clearly, it seems possible to combine non-referentiality and non-anaphoricity detection. This is also shown by de Marneffe et al. (2015) and their work on what they call 'singleton detection'. Since the goal is to maximize the improvement in coreference resolution performance, a mention filtering task that has the largest scope possible is perhaps preferable for this project.

1.3.3 The effect of mention filtering on coreference resolution

Since mention detection is not a directly useful task by itself, but rather a 'helper task' for coreference resolution, an important aspect of quantifying the merits of a mention filtering system is to assess its effect on coreference resolution systems. Looking at previous work in which mention detection systems were evaluated in-line with coreference resolution systems should provide useful insights in what makes a mention detection system effective. Also, it sheds some light on an aspect of mention filtering that is often overlooked, namely how to integrate mention filtering with coreference resolution systems.

Uryupina (2009) describes what is essentially the same system as Uryupina (2003), using a support vector machine instead of Ripper, but applied to non-antecedent rather than non-anaphoric mention filtering. The performance on non-antecedent filtering reaches 69% precision at 96% recall, which is only slightly above a baseline that classifies all NPs as non-antecedental. Since performance is not that high, integrating the filtering into a coreference resolution system does not improve its performance, since it only boosts precision slightly, at a large recall cost.

Nevertheless, Uryupina (2009) provides useful information on the potential of discourse-new and antecedenthood classification. She implements a 100% accurate 'oracle' classifier as part of the coreference resolution system. Adding perfect discourse-new filtering improves the coreference resolution F-score by approximately 3.5%, non-antecedent filtering by 5% and adding both by 9%, in all cases by strongly boosting precision at a relatively small recall loss. This is testament to the potential that mention filtering has to improve coreference resolution performance.

Another look at the relation between mention filtering and coreference resolution performance is provided by Ng and Cardie (2002) and their supervised machine learning system for discourse-new mention filtering. Using the C4.5 decision tree classifier (Quinlan, 1993) and a set of lexical, syntactical, semantic and positional features, they achieve 85% accuracy in discourse-new classification on the MUC-corpus.

Incorporating this system in their own coreference resolution system does not boost its performance, but hurts it, which is similar to what Uryupina (2009) found. However, by preventing the discourse-new filter from being applied to NPs for which two high-precision coreference resolution features indicate that it should not be filtered out, they manage to reduce the impact of overzealous filtering. After this improvement, the integration of the discourse-new filter boosts the F-score of the coreference resolution system by 2.5%. To put this into context, they report that a 100% accurate discourse-new filter would improve the coreference resolution F-score by 9%.

Clearly, the way in which a filter is utilized makes a large difference to its effectiveness. Here, making the application of the filter more restrictive boosts performance. This shows the value of testing different ways of incorporating mention filtering output, and several options are explored in the current project, cf. Section 2.5.

The difference between the effect of the actual mention filtering systems and the oracle systems indicates that there is a substantial overlap between the NPs that are problematic for mention filtering systems and those that are problematic for coreference resolvers. In other words, they seem to cover a lot of the same ground. The Ng and Cardie paper, for example, shows that the 85% of NPs accurately classified by the discourse-new detector contribute only 2.5% in F-score improvement on the coreference task, while discourse-new classification of the remaining 15% adds another 6.5% to the coreference F-score. This is confirmed by the findings of Uryupina (2009).

Byron and Gegg-Harrison (2004) express a similar idea, namely that the NPs filtered out by their mention filtering system are mainly those NPs that are not included in coreference chains by the pronoun resolver anyway. Based on this, we can draw the conclusion that mention filtering systems seem to pick the same low-hanging fruit that can be reached by coreference resolution systems, too. On the one hand, this can be taken as a reason to make an effort to improve other parts of coreference resolution systems, rather than mention filtering. Here, we are more optimistic, and take it to mean that further improvements made on mention filtering systems are likely to have the largest effect on coreference resolution performance, and therefore are well worth the effort.

Another source of information on the effect of mention filtering on coreference resolution comes from Poesio, Alexandrov-Kabadjov, Vieira, Goulart and Uryupina (2005), which is mostly interesting because it reports a positive effect of mention filtering, which contradicts the works mentioned earlier. They propose a comprehensive system, using an SVM, the C4.5 decision tree algorithm and a multi-layer perceptron (MLP, a type of neural network). The MLP outperforms the other systems, and using a set of features concerning modification, text position, superlatives, Uryupina's definiteness probabilities, proper names and predicates, they achieve an F-score of 90% and an accuracy of 85% on discourse-new classification for definite mentions.

That the multi-layer perceptron out-performs other classifier types shows that neural networks can be useful for mention detection. This is testament to the generalization power of multi-layer perceptrons, and indicates their value as a machine learning methods. However, the type of feature used by Poesio et al. are unrelated to the semantic word vectors used in this work.

The effect of this filter on their coreference resolution performance is positive, yielding a 3 percentage point increase in precision, recall and F-score. This contrasts with the findings of Ng and Cardie (2002) and Uryupina (2009), but can be explained by the fact that Poesio et al. use a less advanced resolution system, which benefits more from filtering. Byron and Gegg-Harrison (2004) report a similar effect, where their filter benefited a simple baseline pronoun resolver, but not a more advanced system.

The state-of-the-art in mention detection and filtering is the system described by de Marneffe et al. (2015), which makes it the most relevant for comparison to the current project. As said earlier, de Marneffe et al. introduce singleton detection as a task and build a successful singleton detection system. They build a logistic regression model, that utilises two types of features. The first type of features is based on the discourse-theoretical work by Karttunen (1976), similar to the work of Byron and Gegg-Harrison (2004). The second type is a set of surface features that is similar to those used by Bergsma and Yarowsky (2011), including noun type, animacy, person, number, NE-type, positional, grammatical and semantic features.

The discourse-theoretical observations are captured by features that are based on semantic cues in the environment of the mention. For example, de Marneffe et al. observe that mentions under the scope of modality and negation are more likely to be singleton, and that this effect is strongest for indefinite NPs. This is captured by a feature that indicates whether a mention is under the scope of modality or negation, and a combination feature of modality/negation and the definiteness of the mention.

By looking at coefficient estimates in their logistic regression model, they evaluate whether these features have the expected effect on singleton probability. They conclude that this is almost always the case, which confirms their observations. In addition to features concerning modality and negation, there is also a feature that captures whether the mention is under the scope of an 'attitude verb', and combination features based on that.

The surface features used are fairly typical of those used in earlier coreference resolution and mention filtering work. They find, for example, that animate mentions are more likely to be coreferential than inanimate mentions. They also find that certain types of named entity (sums of money, quantities, percentages, ordinal numbers, and nationalities/religions) are more likely to be singleton. Positional features indicate whether a mention is the first or last word in the sentence, part of a coordinating structure, and its syntactic relation in the sentence.

The performance of their system is evaluated both stand-alone on the OntoNotes data (version 5.0, Weischedel et al., 2013) and in-line with the Berkeley (Durrett & Klein, 2013) and Stanford (Lee et al., 2013) coreference resolution systems, on both the CoNLL-2011 and 2012 tasks.

Using the model that combines both sets of features, de Marneffe et al. report 81% recall and 81% precision on singleton detection. Using a model that emphasizes precision, they reach 56% recall and 90% precision on the same dataset. For the Stanford system, they report an increase in CoNLL F-score of 0.5-1.3 percentage points, and a 0.6-2.0 percentage point increase for the Berkeley system.

De Marneffe et al. show that, using both low-level surface and linguistically-informed high-level features, a high-performance singleton detection system can be built and that even the state-of-the-art in coreference resolution can benefit from a singleton detection system, given that it is of high quality and focuses on precision.

All in all, mention detection is a task that poses many challenges for any automatic solving attempt. As has been shown, there are many tasks that can be grouped under the header of mention detection or filtering, all with different challenges. In this thesis, the singleton detection task will be tackled, rather than any of the other tasks. Singleton detection has numerous benefits. It encompasses all other mention detection tasks, since it covers both anaphoricity and coreference. It has shown promising results for benefiting coreference resolution. In addition, it is well-suited for evaluation in-line with coreference resolution systems on the CoNLL task, and it is clearly defined and easy to operationalize, cf. de Marneffe et al.'s definition: *"any kind of non-referential NP as well as any referential NP whose referent is mentioned only once (i.e., singleton)"* (de Marneffe et al., 2015).

1.3.4 Mention detection and filtering in coreference resolution systems

Apart from the mostly stand-alone attempts at various forms of mention detection discussed in previous sections, it is also interesting to take a look at how existing, high-performance coreference resolution systems deal with mention detection and filtering.

In order to get an overview of the importance of and approach to mention filtering in state-of-the-art coreference resolution systems, the participating systems in the CoNLL-2011 and 2012 Shared Task (Pradhan et al., 2011, 2012) are discussed, in addition to the current best-performing coreference resolution system, the Berkeley Coreference system (Durrett, Hall & Klein, 2013; Durrett & Klein, 2013).

The CoNLL-2011 Shared Task was a coreference resolution task, using data from the OntoNotes corpus (version 4.0, Weischedel et al., 2011). This corpus includes, in addition to coreference annotation, several other annotation layers: syntax trees, verb and noun propositions, word senses, and named-entity types.

The task is to perform coreference resolution (i.e. predict the gold-standard coreference annotation layer) on the English-language part of the corpus, which is then scored using five different metrics: MUC (Vilain, Burger, Aberdeen, Connolly & Hirschman, 1995), B-CUBED (Bagga & Baldwin, 1998), CEAF$_m$, CEAF$_e$ (Luo, 2005), and BLANC (Recasens & Hovy, 2011). The MUC-metric focuses on the linking of pairs of mentions, B-CUBED captures how well mentions are assigned to entities, the CEAF-metrics are entity-based measures, and BLANC is a version of the Rand index (Rand, 1971), adapted for coreference. The average of the MUC, B-CUBED and CEAF$_e$ F1-scores is used as a final score, the CoNLL F1-score.

The CoNLL tasks' results report scores for 'mention detection' for participating systems, but these cannot be used to compare actual mention detection performance of the systems. Mention detection scores are calculated based on the final output. In the final output, all singleton mentions have been filtered out, because these are not annotated in the OntoNotes corpus. Therefore, the mention detection score is very much determined by the clustering part of the coreference resolution system, rather than its mention detection and filtering qualities. Pradhan et al. describe this as follows: *[The systems will] not get credit for the singleton entities that they correctly removed from the data, but they will be penalized for the ones that they accidentally linked with another mention.".* Therefore, it is only interesting to look at the approaches used by shared task participants, rather than their mention detection scores.

Another useful part of the CoNLL-2011 task is that it initially uses automatically generated annotations and no mention information, but that it also tests performance using gold-standard annotations, gold-standard mention spans (i.e. chunking information) and gold standard mentions (i.e. all and only mentions that are part of coreference chains). The significance of this is that the performance of systems using gold standard mentions provides an upper bound on the performance if mention detection were perfect. In addition, it is a good measure of performance for the clustering part of the coreference resolution systems.

First, the performance with and without gold-standard mention information will be compared, to get an idea of the capabilities of state-of-the-art coreference resolution systems and to gauge the maximum possible impact that mention detection systems can have. The tests using gold-standard mention boundaries will not be considered, since they were found to have only a minor effect on coreference resolution performance. After that, we will consider the types of mention filtering approaches used.

In the CoNLL-2011 task, only 2 out of 23 participants evaluated their systems using gold-standard mentions. The Lee et al. system had an F-score of 58.31 in the original condition, which increased to 73.05 using gold-standard mentions. The Chang et al. system received an F-score of 73.83 using gold-standard mentions, as compared to an initial 55.96. Clearly, increases of 15 and 18 percentage points should be taken as an invitation to improve mention detection and filtering systems, while also showcasing the relative quality of the clustering parts of these systems. These results match the findings of Uryupina (2009) and Ng and Cardie (2002), who also found large increases in coreference resolution performance when using 'perfect filtering' of mentions, albeit in a slightly different way.

The CoNLL-2012 Shared Task is identical to the CoNLL-2011 task, except for its inclusion of the Chinese and Arabic-language parts of the OntoNotes corpus in addition to the English part. In this task, a larger proportion, 8 out of a total of 16 participants evaluated using gold-standard mentions. Since most of these systems also tested on all three languages involved, this yields a lot more data than the 2011 task. Generally, though, the picture is congruent with that of the year before: gold-standard mentions significantly improve coreference resolution performance. Clearly, the size of the effect depends strongly on the system, with some only gaining 6 percentage points in F-score (Fernandes, dos Santos & Milidiú, 2012), while others gain up to 17 percentage points (Chang, Samdani, Rozovskaya, Sammons & Roth, 2012).

Discussing the effect of mention filtering, Pradhan et al. note that recall is the most important factor in mention detection, since the clustering part of the system cannot recover from missing mentions, while it can often deal properly with incorrect mentions. This means that a mention filtering system should focus on achieving high-precision, i.e. rather filter out a smaller number of incorrect mentions than filter out a larger number of incorrect mentions, at the cost of excluding a higher proportion of correct mentions.

Looking at how mention detection was implemented in coreference resolution systems, we see that in the CoNLL-2011 task, only one system, Cai et al. (2011), attempted joint mention detection and coreference resolution. All other systems did these two things independently. Most participants did not attempt what can be properly called mention filtering: they simply considered all NPs as mentions, basing their selection on POS and NER tags, sometimes with some additional selection heuristics.

Only 4 out of 23 systems used either a rule-based or a machine-learned non-referential *it* filter. In addition, some used heuristics to remove mentions like numeric entities, which has more to do with the OntoNotes guidelines than with mention filtering per se. Only one system, that of Song, Wang and Jiang

(2011), uses a machine-learned mention detection classifier. Using a maximum entropy (MaxEnt) classifier and a set of lexical, POS, positional, semantic, NER and NP-type features, they achieve a mention detection performance of 53% recall, 81% precision and 64% F-score. It is not clear whether this is a true mention detection score, or the one calculated as in the CoNLL task, which depends strongly on the rest of the coreference resolution system.

In the CoNLL-2012 task, we find more advanced mention filtering approaches. Oddly enough, the best scoring system (Fernandes et al., 2012) does not do any mention filtering, instead it simply selects all noun phrases, pronouns and named entities , and likewise do 6 other systems (some with language-specific adaptations). Of the remaining 9 systems, 1 has a non-referential pronoun filter, 4 have a non-referential *it* filter and 4 do some form of singleton detection.

Approaches vary widely: one of the highest-scoring systems, by Björkelund and Farkas (2012), uses a MaxEnt classifier and an extension of the feature set used by Boyd et al. (2005) to filter out non-referential *it*, *we* and *you*, while another high-scoring system, that of Chen and Ng (2012), attempts singleton detection by including the proportion of singleton occurrences of the NP's head noun as a feature. Notably, three different systems use (an adaptation of) the rule set used by Lee et al. (2011), which is a set of regular expressions that capture some of the most common patterns that include non-referential *it*.

In general, the mention filtering systems are only briefly described in the system papers, and those who test the performance of their system with and without these filters conclude that the effect on resolution performance is small. This contrasts with the sizeable effect seen when gold-standard mentions are used, which serves to illustrate the performance gain that is still to be made by the further development of mention filtering systems.

The lack of importance given to mention detection in resolution system's development is further substantiated when we look at the current best-performing corefernece resolution system, the Berkeley Coreference System (Durrett et al., 2013; Durrett & Klein, 2013). This system, remarkably, does not filter any mentions, it simply selects all candidate mentions based on NE- and POS-tags, and the authors state the following: *"[W]e aim for the highest possible recall of gold mentions with a low-complexity method, leaving us with a large number of spurious system mentions that we will have to reject later."* However, as de Marneffe et al. (2015) have already shown, this does not mean that their system does not benefit from improved mention filtering. The Berkeley system's CoNLL F-scores improved by 0.6-2.0 percentage points when combined with de Marneffe et al. singleton detection system.

System	Classification method	Feature types	Task	Corpus	Performance
Paice (1987)	Hand-crafted rules	-	Non-referential *it* detection	Technical documents	Acc-87%
Evans (2001)	Memory-based learning (MBL)	Lexical, part-of-speech, pattern-like and positional	Binary non-referential *it* detection and 7-way classification of *it*	SUSANNE, BNC	Binary: Acc-71%, 7-way: P-73%, R-69% for non-ref. *it*
Litrán (2004)	Support-vector machine (SVM), MBL	Lexical, part-of-speech, pattern-like and positional	Non-referential *it* detection	Biomedical abstracts	Acc-93%
Boyd (2005)	MBL, decision tree	Part-of-speech, hand-crafted linguistic patterns	4-way non-referential *it* detection	BNC Sampler	Acc-88%, P-82%, R-71%
Bergsma (2011)	Logistic regression	Lexical, N-gram counts, syntax, positional, surface semantic	Non-referential *it* detection	BBN	Acc-86%, P-82%, R-63%
Byron (2004)	Hand-crafted rules	Part-of-speech, surface semantics, discourse patterns	Non-referential indefinite NP detection	Penn Treebank/WSJ	Filter out 12% of NPs, no P/R/Acc.
Bean (1999)	Hand-crafted syntactic heuristics	Syntax, positional, definiteness probabilities	Non-anaphoric definite NP detection	Newswire text	82%-P, 82%-R
Uryupina (2003)	Automatic rule induction	Syntax, part-of-speech, head word, definiteness probabilities	Non-anaphoric NP detection	MUC-7	89%-P, 84%-R
Uryupina (2009)	SVM	Syntax, part-of-speech, head word, definiteness probabilities	Non-antecedental NP detection	MUC-7	69%-P, 96%-R
Ng (2002)	Decision tree	Lexical, syntax, semantic, positional	Discourse-new NP detection	MUC-6/7	Acc.-85%
Poesio (2005)	SVM, decision tree, multi-layer perceptron	Surface semantics, positional, definiteness probabilities	Discourse-new definite NP detection	GNOME	Acc.-86%
Marneffe (2015)	Logistic regression	Lexical, N-gram counts, syntax, positional, surface semantic, discourse patterns	Singleton detection	OntoNotes 4.0	81%-P, 81%-R or 90%-P, 56%-R

Table 1: An overview and summary of existing mention detection systems.

1.4 A new approach: semantic word vectors and neural networks

1.4.1 Lessons for singleton detection

Essentially, a perfectly performing (i.e. 100% precision, 100% recall) singleton detection system would be equivalent to the gold-standard mention condition of the CoNLL tasks. This provides a clear upper bound on the influence of the singleton detection system, and marks the limits of what a mention filtering system should cover.

When considering the singleton detection task, the first thing to be noted is that the notion of a perfect, independent singleton detection system unrealistic. By definition, singleton detection amounts to knowing which mentions are part of a coreference cluster. It is safe to assume that this cannot be done with 100% accuracy without knowing, in some cases, which other entities are part of the coreference cluster. The knowledge of which mentions form a coreference cluster is the task of the other part of the coreference resolution system, and thus an independent system can never perform perfectly. Compare, for example, the two instances of *the cat* in Examples 18a and 18b. Assuming the story ends there, there is no way of knowing from the direct context whether *the cat* here is singleton or not.

(18) a. [A boy]$_1$ chased [the cat]$_2$ into the room. I picked up [the little fluffball]$_2$ (and kept him safe).
b. [A boy]$_1$ chased [the cat]$_2$ into the room. I picked up [the kid]$_2$ (and made him stop).

One way of dealing with this is to accept it and see singleton detection as a initial filtering step that makes the rest of the resolution process faster, simpler (less mentions to consider) and more accurate. An example of this is the filtering implemented in the Stanford coreference resolution system by de Marneffe et al. (2015).

Another approach is to somehow integrate the two parts. Examples of this are Denis and Baldridge (2007) and Cai et al. (2011), who propose systems that do mention detection and coreference resolution jointly. Another way to integrate the two is presented in Poesio et al. (2005), who did a pre-classification of mentions as direct anaphora, after which their discourse-new classification was applied, which in turn was followed by the rest of the resolution process. Similarly, Ng and Cardie (2002) applied a resolution heuristic before filtering to mitigate the effects of filtering too much mentions out. A last, less complex option is used by de Marneffe et al. (2015), who combine their singleton detection with the Berkeley system by including the output of the first, the probability of a mention being singleton, as a feature for the latter.

The question that lies at the core of singleton detection is: 'Will this entity be referred to later in the text, or not?'. This shows that the problem is ultimately at the level of discourse. Most models discussed earlier focus on syntactic features and patterns and lexicalized surface features. These work reasonably well, but, by their nature, cannot capture the semantics of mentions and sentences. Ultimately, one would like to model semantics above the sentence-level, the realm of discourse, to 'solve' the problem of singleton detection.

This is, of course, not an original insight. Both Byron and Gegg-Harrison (2004) and de Marneffe et al. (2015) attempt to incorporate discourse-level features in their systems, basing their features on the ideas of Karttunen (1976). These features are based on information about modality, negation and attitude verbs. However, in both cases, these features seem to have only limited effect. In the Byron and Gegg-Harrison paper, the system does not significantly improve coreference resolution. The discourse features of the de Marneffe et al. system work very well, but are surpassed in performance by a system using the type of syntactic and surface features mentioned earlier. Nevertheless, the system combining the two types of features outperforms both, indicating that higher-level features are certainly valuable (and feasible) for singleton detection.

In addition to the ideas that singleton detection is limited as an independent module and that higher-level features are necessary to push performance higher, a third notion that is expressed in previous work is that filtering out coreferential mentions hurts performance more than not filtering out singleton mentions. I.e., in singleton detection, precision (filtering out only singletons) is more important than recall (filtering out all singletons). Conversely, in mention detection (not the filtering, but the identification of mentions), recall (including all coreferential mentions in clustering) is more important than precision (having only coreferential mentions as clustering input).

As such, a good singleton detection system should show high precision, with recall mattering less. At the least, it should be able to vary the trade-off between precision and recall, as is possible for example by varying the threshold in a logistic regression classifier.

During the design of a singleton detection system, it should also be kept in mind that a large proportion of the non-referential pronouns occur in fixed constructions, patterns and idioms. These patterns are described and tested in, among others, the works of Paice and Husk (1987) and Boyd et al. (2005). Examples are non-referential *it* as it occurs with weather verbs (e.g. Example 12) or with certain fixed expressions (e.g. Example 17). Any system that does not have these patterns predefined, but rather automatically learns to detect singletons, should somehow show that it manages to learn these patterns to some extent, in order to get good performance.

A final lesson to be taken from previous research concerns the relation between mention filtering performance and coreference resolution performance. It turns out that, in many cases, the mention filtering performance is good when tested independently, but the effect when tested in-line in a coreference resolution system is somewhat disappointing.

The main reason for this is, as suggested by Byron and Gegg-Harrison (2004), that the mentions that are problematic for mention detection are the same ones that are problematic for coreference resolution. I.e., the part of the task that is solved by mention filtering is a part that is mostly also covered by the clustering phase of coreference resolution. This is not all that surprising if one considers the fact that both types of systems tend to use the same types of features.

Put otherwise, two systems that use the same information for related tasks, are likely to make similar mistakes and successes. A new singleton detection system should therefore aim to make gains in areas that benefit the clustering phase the most, possibly by considering different features.

1.4.2 Why neural networks and semantic word vectors?

All in all, these 5 things should be kept in mind when developing a singleton detection system:

1. It is limited as an independent module
2. To push performance higher, features covering more than morphosyntax are needed
3. Precision in filtering is more important than recall
4. A good system should be able to capture highly frequent patterns
5. It should focus on ground not already covered by coreference resolution systems in order to have an effect.

As mentioned earlier, the hypothesis is examined that these considerations can be addressed in order to build a state-of-the-art singleton detection system by using semantic word vectors and neural network architectures.

Semantic word vectors (also called neural word embeddings) are a way of representing words that has been around since the 1990s, and have gained much traction in recent years. Words are represented as real-valued dense vectors in a high-dimensional continuous space. These vectors specify a position in the high-dimensional space, which specifies the semantic value of a word with regard to all the dimensions.

The vectors encode differences, similarities and relations between words. They have been shown to capture, for example, the similarity in the relation between *'king'* and *'queen'* and *'man'* and *'woman'* (Mikolov, Yih & Zweig, 2013). In addition, they have been used in state-of-the-art systems for a range of NLP tasks, e.g. POS-tagging, NP chunking, NER-tagging, syntactic parsing, and sentiment analysis (Collobert et al., 2011; Socher, Lin, Ng & Manning, 2011; Socher, Perelygin et al., 2013).

The nature of these word representations, as high-dimensional, real-valued vectors, makes them very suitable for use with neural networks. Several types of neural networks can be used with these word representations, ranging from simple multi-layer perceptrons to recursive, recurrent, convolutional and tensor networks (cf. Section 1.4.4 for details). They constitute a powerful machine learning framework and are well-suited to deal with this type of high-dimensional, automatically derived numerical feature. Different types of neural nets and their exact workings will be discussed in more detail, later.

One of the reasons for focusing on the combination of neural word embeddings and neural networks is that it is a semi-supervised approach that has shown a lot of promise on several NLP tasks in recent years. Using no, or only few additional features, it managed to approach or outperform the state-of-the-art on syntactic parsing (Socher, Bauer, Manning & Ng, 2013), paraphrase detection (Socher, Huang, Pennington, Ng & Manning, 2011), the capturing of semantic regularities (Mikolov et al., 2013) and sentiment analysis (Socher, Perelygin et al., 2013).

However, most of these are tasks that deal with phenomena on the sentence or syntax level, with sentiment analysis as a notable exception. Given that these features contain more semantic information than, for example, part-of-speech and syntactic features, it would be interesting to see if they can be applied to higher-level tasks, such as singleton detection, which cover phenomena that are above the syntax and sentence level, and lie more in the realms of discourse and semantics.

Thus, one of the goals of this thesis is to investigate how far semantic word vectors can go, i.e. if they can boost performance on higher-level tasks, that, so far, have been far from solved.

The second reason for choosing this method is that it seems to tick the boxes of what a good singleton detection system should be able to do.

First of all, a neural network can use a logistic regression layer for classification, and this generates a singletonhood-probability for each mention. This allows for integration into a feature-based resolution system, like de Marneffe et al. did for the Berkeley system.

In addition, varying the threshold probability value for classifying something as singleton or not, allows for manipulation of the precision/recall trade-off. This makes it easy to increase precision (at the cost of recall), which should help to increase its impact on final coreference resolution performance.

Another requirement is that the system should be able to capture highly-frequent patterns which often contain singletons, such as non-referential *it* with a weather-type verb (Boyd et al., 2005) or *'it is* COGNITIVE VERB-*ed that'* (Paice & Husk, 1987). The advantage of using semantic word vectors is that they can encode generalizations like 'weather verb' or 'cognitive verb', since the semantic vector space clusters similar verbs closer together.

Combining this with a supervised neural network approach, it should be able to learn these patterns, exploiting the similarity between vectors of words that form certain groups. An advantage is that these groups are not limited to fixed lists of words (e.g. a list of cognitive verbs), but capture group membership in a continuous manner.

What remains are the two most important requirements for improved singleton detection: the need for higher-level features and the filtering out of those mentions that are problematic for coreference clustering. As the name implies, semantic word vectors capture semantic similarities between words, in addition to carrying syntactic information. This should cover the information used by previous systems, and add a layer of information not used before.

This additional information should also help to filter out precisely those mentions that are problematic for coreference resolution systems. Generally, coreference resolution systems use largely the same types of features that previous mention detection systems have used. However, no systems make use of semantic word vectors. Thus, adding a new information source to the mention filtering and clustering process should help in tackling part of the problem space that current coreference resolution systems cannot yet cover.

How exactly semantic word vectors can work as features for singleton detection is hard to pinpoint, especially when compared with more transparent features, such as the patterns as defined by Paice and Husk (1987). Nevertheless, we can look at the information that is contained in semantic word vectors, and argue how that information could be relevant for a singleton detection model. In order to do this, it should be noted that features or vectors do not indicate whether something is singleton or not. Rather, they positively or negatively influence the predicted probability of whether a mention is singleton, which is a small but important difference.

The main strengths of semantic word vectors are that they capture similarities between words and relations between words. In the vector space, similar concepts are grouped together, such as country names. In addition, they capture relations between words, such as between male and female counterparts of the same word, or between countries and their capitals.

The knowledge about similar words can be used in a way similar to many of the pattern-based features we have seen in previous work. We know, for example, that *it* followed by a cognitive verb, or by *is* and a word that indicates a state, is more likely to be non-referential. Similar to the clusters depicted in Table 3, we can imagine a cluster of status words, e.g. *likely, possible, impossible, probable, unbelievable.* Training the neural network would then result in the network assigning a higher singleton probability to an occurrence of *it* followed by *is* and one of these words.

Of course, the same can be achieved using lexicalized features, but this would require a feature for each of these words, and a number of training examples for each feature in order to learn to make use of the features. The advantage of semantic word vectors is that they already contain the information that these words are similar. When the neural network is confronted with a training example containing one of these words, it will learn in such a way that it will also affect its learning of examples containing the other words in the group. This means that it needs fewer examples to acquire the 'pattern' containing any of these words, and that it can even generalize to unseen words, as long as they are part of the semantic word vector space.

Another way in which the similarities captured by semantic word vectors can help in predicting singleton probability is by identifying whether a mention is central or peripheral to the story. For example, if a mention is semantically similar to other mentions in the text, it is likely to part of the central 'story' of the text, and thus more likely to re-occur in the text. This, in turn makes it less likely to be singleton. In the opposite way, semantically dissimilar mentions are more likely to be singleton.

Similarly, semantic word vectors can cover the information that has been captured by other morpho-syntactic and surface semantic features in previous work. Looking at the model of de Marneffe et al. (2015), for example, we see that they find that certain types of named entities, such as languages and geographical locations, are less likely to be singleton. For some other types of named entities, the reverse is true, and other aspects of the mention, such as animacy, number and gender are all found to affect singleton probability in their own way, too. We posit here that semantic word vectors capture all this information as well. See, for example, the first column of Table 3, which provides the same information as is otherwise indicated by a geographical-location-type named-entity-tag.

A point where semantic word vectors deviate from other commonly-used features is that they contain information about relations. Although less directly than the information on similarity, knowledge of how words relate to each other can also be useful for singleton detection. Take, for example, a case where, in a newswire text, a company and its CEO are named, in addition to another person. In this case, because the company and the CEO are related, it is more likely that they are the topic of the text than the other person. This, in turn makes the likelihood that they are singleton lower, since they are more likely to be referred to later in the text.

It should be noted, however, that the fact that this information is somehow contained in semantic word vectors does not mean that this information is also used by the singleton detection model. Whether this knowledge can be maximally exploited to create a high-performance singleton detection model depends on the architecture that the semantic word vectors are used in. In this case, it depends on choosing the right type of neural network, the right structure, the right training methods, and so on. This, in addition to the general difficulty of interpreting neural network models, makes it difficult to assess how and to what extent the 'reasonings' described above are actually used.

1.4.3 Semantic word vectors

There are three main ways in which semantic word vectors can be used for NLP: they can be initialized randomly and trained during the task, they can be pre-trained on a corpus of raw text, and then used as is, or they can be pre-trained on raw text and trained during the task, too. Here, we will discuss all three options, by focussing on two approaches to semantic word vector training, by Collobert et al. (2011) and by Pennington, Socher and Manning (2014).

Collobert et al. used a text collection of approximately 850 million words of normalized and tokenized text for the training of their vector space model. Semantic word vectors can be trained using various unsupervised learning methods, and here, a neural network was used.

One way of doing this is to use a pairwise-ranking approach, in which the goal is to assign a higher 'score' to a sentence (or rather a n-word window) from the corpus than to the same sentence with one word replaced by a random word. Vectors are initialized randomly, and backpropagation is used to train the neural network so that it learns to do the pairwise-ranking task well.

This is done for the whole corpus, and the result is a set of semantically and syntactically conditioned vector representations of words. Instead of learning a representation for each word in the corpus, a fixed-size vocabulary of the most frequent words is used, and the rest of the words are collapsed under an 'unkown word'-token.

Relevant for the choice of vectors in this project is that Collobert et al. trained two vector sets, each on a different size corpus. Both were used in a range of NLP tasks, and they concluded that the language model based on the larger corpus showed better performance.

Pennington et al. (2014) use a different, more advanced approach to the learning of word vectors. They aim to avoid the disadvantages of both methods that use local context windows (like the one above) and methods based on global matrix factorization methods. The latter attempt to capture the information of large matrices of corpus statistics, such as word co-occurrence statistics, and reduce them to a lower-dimensional vector space.

The model by Pennington et al. is called Global Vectors (GloVe) and is closest to the global matrix factorization family of methods. They use a global bi-linear regression model to efficiently capture the information from a word-word co-occurrence matrix over the whole corpus. They use very large corpora, ranging from 1.6 to 840 billion words, and use vocabularies containing 400K to 2M words. They measure performance on a set of NLP tasks, and find that, again, that vectors trained on a bigger corpus yield better performance than those trained on a smaller corpus. They also find that higher-dimensional vectors, e.g. 300-dimensional ones, perform better than smaller, 50-dimensional vectors.

Pennington et al. compare their vector space model directly to that of

Collobert et al. on an NER-task. They find that, using vectors of similar dimensionality, that their vectors outperform those of Collobert et al. by 1 to 2 percentage points in terms of F-score. It is unclear whether this is due to the model used for creating the word vectors, or simply due to the use of a larger corpus.

Because of the quality of the word embeddings produced by these methods, it is very much possible to use them as they are for NLP tasks. This is one of three ways of using semantic word vectors, with pre-trained vectors and without task-specific training. The two other options are to only do task-specific training, starting with random vectors, or to do both pre- and task-specific training.

In the case of task-specific training, the word vectors are not just seen as features to help the task, but also as parameters that can be optimized. Using a neural network to do sentiment analysis, for example, the goal during training is to optimize the model's accuracy in predicting the correct sentiment. Normally, this is done by backpropagating the error on the predictions, and adjusting the weights of the neural network accordingly. In task-specific word vector training, however, the error is not just backpropagated to the weights, but also to the input, i.e. to the semantic word vectors.

In the case of pre-trained vectors, this keeps the general geography of the vector space intact, but adapts it so that the vector representations are more specifically suited for the task in question. In the sentiment analysis task, for example, this helps to cluster sentimentally similar words more closely and move positive and negative words farther from each other. Conversely, when done for a syntactic parsing task, the vector space might be changed so that syntactically similar words, e.g. modal verbs, are closer together.

In the case where only task-specific training is done, we start with randomly initialized vectors, and the vector space is then trained to be optimally suitable for the task. Using enough labelled data and supervised training, this works well, but it misses out on a lot of background knowledge that is gathered by pre-training on very large corpora. This difference is illustrated very clearly by Collobert et al. (2011), who train and test using both randomly initialized and pre-trained vectors. They show that pre-training significantly, and sometimes quite drastically, improves performance of their convolutional neural network (CNN) on POS-tagging, chunking, NER and semantic-role labelling (SRL) tasks.

In addition, they illustrate the changes made to the word vectors nicely by comparing the nearest neighbours in the vector space for a random set of words. Table 2 shows the ten nearest neighbours for six of those words without pre-training, Table 3 shows the same thing, after pre-training. As can be seen, the nearest neighbours in Table 3 are very closely related to the word at the top of the column, morphologically, syntactically, as well as semantically. The same cannot be said for the words in Table 2, which seem, at least to the human eye, random. Clearly, the pre-training transforms the vector space to

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
PERSUADE	THICKETS	DECADENT	WIDESCREEN	ODD	PPA
FAW	SAVARY	DIVO	ANTICA	ANCHIETA	UDDIN
GIORGI	JFK	OXIDE	AWE	MARKING	KAYAK
SHAHEED	KHWARAZM	URBINA	THUD	HEUER	MCLARENS
RUMELIA	STATIONERY	EPOS	OCCUPANT	SAMBHAJI	GLADWIN
PLANUM	ILIAS	EGLINTON	REVISED	WORSHIPPERS	CENTRALLY
GOA'ULD	GSNUMBER	EDGING	LEAVENED	RITSUKO	INDONESIA
COLLATION	OPERATOR	FRG	PANDIONIDAE	LIFELESS	MONEO
BACHA	W.J.	NAMSOS	SHIRT	MAHAN	NILGIRIS

Table 2: The ten nearest neighbours in vector space by Euclidean distance for the top word. Vector space trained with random initialization on an SRL task. From Collobert et al. (2011).

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Table 3: The ten nearest neighbours in vector space by Euclidean distance for the top word. Vector space trained on a large corpus of 600M words, using the method by Collobert et al. described above. From Collobert et al. (2011).

a seemingly more interesting and more meaningful one, capturing background knowledge that is essential for natural language understanding.

An interesting aspect of the Collobert et al. vectors can be seen by looking at the column starting with REDDISH in Table 3. Here, it shows that the word embeddings carry a very strong syntactic element, rather than capturing a semantic similarity. Under REDDISH, one might expect, for example, also RED, PINK or ORANGE. However, we find only '*-ish*'-colours, with no apparent ordering in the colours. Apparently, the cluster around REDDISH is based more on morphological characteristics than on the semantics of colours.

This is not necessarily problematic, but it does emphasize the point that different word vectors, or task-specifically trained word vectors might have different benefits for certain tasks.

On the other hand, task-specific training diminishes one of the strong points of semantic word vectors, namely that they can be used for a range of NLP tasks off-the-shelf. The vectors carry a lot of generally applicable, linguistic information, and task-specific training makes them more like task-specific features. This makes them harder to use within other frameworks, or in an NLP system that attempts to do more than one task. Rather, it might be better to use higher-dimensional vectors, trained on larger corpora. If the vectors have more dimensions, they can capture more specific information, which might boost task performance without sacrificing generalizability.

1.4.4 Neural networks for NLP: an overview

Neural networks, often under the moniker of 'deep learning', have made great advances in recent years, for example in the field of computer vision. More recently, these approaches have been extended to tasks in the field of natural language processing. Combined with semantic word vectors, they have shown great potential and results. Here, a number of these systems and their underlying neural network architectures will be briefly reviewed, to get an overview of the available approaches to singleton detection.

The most basic type of neural network is the multi-layer perceptron (MLP). The multi-layer perceptron consists of three or more layers, an input layer, one or moree hidden layers and an output layer. Each layer consists of a certain number of nodes, and each node gets input from every node from the layer below and outputs to every node in the layer above. So, a node in the hidden layer gets input from every input node, and the hidden node outputs to every node in the output layer. In a node, all inputs are combined by multiplying them by a set of weights. Then, an activation function is applied to this value to provide the output of the node.

The output layer, also called classification layer, is used to generate a final output value, given a certain input. In singleton detection, for example, a logistic regression layer might be used for the output layer, which then outputs a probability that the mention in the input is singleton.

The network is trained in a supervised manner. This means that the output probability is compared to the label of the input (e.g. 0 for singleton), and an error value is computed. Using a training method called backpropagation, this error value can be used to update the weights of the MLP so that the error is minimized for subsequent examples. If enough training examples are provided enough times, this hopefully leads to a neural network that has learned the task in a generalized manner. For a more detailed explanation of neural network structure and training, see Section 2.4.

Although multi-layer perceptrons are a relatively simple framework, they are quite powerful. In principle, they can approximate any function, and additional hidden layers can be used to aid in learning more complex tasks. As is, an MLP can be used with semantic word vectors, or any other vectorized input, to do any NLP-task that requires labelling, classification or tagging.

Because of the structure of the MLP, it learns in a relatively unguided or unbiased manner. It maps the input to the output, without imposing or assuming any structure in the input, since all nodes are connected to all other nodes. This makes it suitable to work with an unordered set of input information, much like many other machine learning methods.

Nevertheless, many recent neural network-based works in NLP have used different, more sophisticated architectures. The main reason for this is that performance is generally better if the architecture is suited to the structure of the problem or task. For example, if one wants to extract certain features repeatedly from ordered input, a convolutional neural network can be used, which works with local feature extractors called feature maps. For language, specifically, researchers have sought to exploit the recursive nature of language. Since language units like sentences have inherent (recursive) structure, a neural network can learn better when they are adapted to work with this recursion, rather than assuming the input is an unordered sequence.

Recursive neural networks (RNN, not to be confused with the recurrent neural network, also abbreviated RNN), are a rather straightforward extension of the multi-layer perceptron. The idea behind RNNs is that the same neural network can be applied to sub-sets of the total input (e.g. a sentence). In the case of a sentence, the recursive NN is applied to two or more input word vectors and generates a single vector of the same size as one word vector. By doing this repeatedly, the whole sentence is eventually covered by one representation. These intermediate and final representations can then be used for various tasks, e.g. for parsing or paraphrase detection.

The ability to combine parts of a tree in a consistent way makes RNNs very suitable for syntactic parsing, as demonstrated by Socher, Manning and Ng (2010), Socher, Lin et al. (2011). Socher et al. (2010) achieve near state-of-the-art syntactic parsing performance on well-known benchmark corpora. Parsing using a recursive neural network is done by representing words from a sentence as semantic word vectors, combining pairs of adjacent words using the hidden layer, and adding a classification layer that produces a score for the parse. Using a treebank, the network can then be trained to maximise the score for correct parsing decisions, resulting in a high-performance syntactic parser.

Remarkably, Socher, Lin et al. (2011) shows that the same algorithm can be used for another type of parsing, that of natural scene images. Similar to the way it constructs a syntactic parse tree from a set of word vectors, it can create a parse tree from a set of segments from an image, and classify segments as well. In both tasks, the recursive neural network out-performs the then state-of-the-art systems. The similarity between the task and the workings of the RNN are nicely illustrated by Figure 1.

Note that the RNN-based parser of Socher et al. only uses word vectors as input, where many other parsers also use part-of-speech information. It is nice and simple to have one general composition function (set of weights/hidden

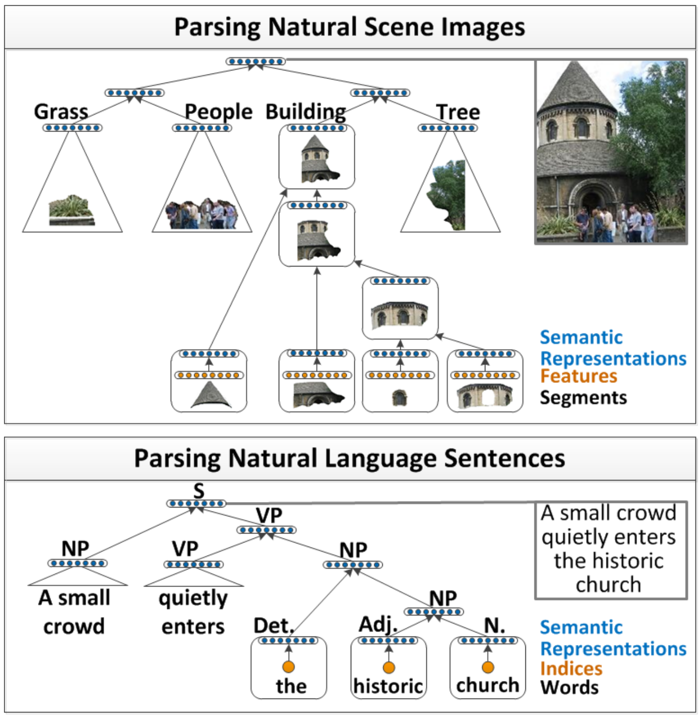

Figure 1: Illustration of a recursive neural network architecture which can be applied to both syntax and image parsing. From Socher, Lin, Ng and Manning (2011)

layer), but it is implausible that the meanings of a determiner and a noun are composed similarly to those of a noun phrase and a verb phrase, for example. Rather, it makes more sense to have different composition functions for different parts-of-speech, or even for different words. This is the reasoning behind the syntactically-untied RNN (SU-RNN), which uses POS-specific weight matrices and the related compositional vector grammar (CVG).

The syntactically-untied RNN, as presented in Socher, Bauer et al. (2013), uses the part-of-speech tags or phrase labels of its input nodes to condition the weight matrix used. So, instead of training one neural net to capture all syntactic composition, a neural network is trained for each pairwise combination of syntactic labels. Interestingly, this shows that these networks can capture linguistic notions too, e.g. the idea of head words. Looking at the weight matrix learned for the combination of a determiner and a noun phrase, for example, it can be seen that the resulting node takes most of its meaning from the NP, not from the determiner.

A further improvement on the SU-RNN is the compositional vector grammar, which is a parsing model that combines a neural network with a probabilistic context free grammar (PCFG) for structure prediction. This helps inform the parse quality by using knowledge of likely parse decisions.

Outside of the family of RNN-based models, there are three other types of neural nets which have been used for NLP, and which should be considered here: (recursive) auto-encoders (e.g. Socher, Huang et al., 2011), recurrent neural networks (e.g. Mikolov et al., 2013), and convolutional neural networks (e.g. Collobert et al., 2011).

An auto-encoder is a type of neural network that is suitable for unsupervised learning. The auto-encoder is similar to the MLP, but instead of having a classification layer as output, it has an output layer that attempts to reconstruct the input as faithfully as possible. The main advantage of this is that it can be used to learn compressed representations of inputs. Because the input has to be reconstructed, the network learns to keep as much information in the intermediate representation (the hidden layer) as possible. Because of this, the hidden layer comes to contain a compressed representation of the input. Combined with word vectors, for example, it is suitable for making composed representations of phrases or sentences, with minimal loss of information. These compressed representations can then be used as features for a classification layer or for another neural network.

Similar to the way a neural net can be applied recursively to a structure to create a recursive neural network, so can an auto-encoder be applied to a structure, e.g. a parse tree, to create a recursive auto-encoder (RAE). Socher, Huang et al. (2011) use a recursive auto-encoder approach to paraphrase detection. For every node in the parse tree, a representation is generated, and the representations of each node of the sentence and the possible paraphrase are compared, in order to classify the sentences as paraphrases or not. The RAE approach outperforms previous paraphrase detection systems on a benchmark paraphrase corpus.

For the recurrent neural network family of networks, Mikolov et al. (2013) provide a good example of how they can be used for NLP. A recurrent NN is, in its simplest form, very similar to a MLP, the one addition being an additional hidden layer. This additional hidden layer contains a copy of the regular hidden layer in its previous state; it serves as a kind of memory. The output of this additional hidden layer is fully connected to the input of the regular hidden layer. So, in addition to the regular input, the hidden layer also gets its previous state as additional input.

The 'memory' makes it naturally suitable for tasks dealing with sequences, such as speech recognition (Mikolov, Karafiát, Burget, Černocký & Khudanpur, 2010) and language modelling (Mikolov et al., 2013). Given a word in a sentence as input, for example, it will implicitly take into account the previous word(s), improving its language modelling capability. Mikolov et al. prove this by showing that their method achieves state-of-the-art performance on speech recognition.

Finally, convolutional neural networks (CNNs) are another variation of the basic neural net, and make use of one or more convolutional layers. A convolutional layer is a type of hidden layer which can be put directly after the input layer, and works as a feature extractor. Instead of having all input nodes connected with all convolutional nodes, each convolutional node is connected only to a window of n input nodes. All convolutional nodes have the same weights, which comes down to having the same feature extraction method applied to each window.

The advantage of a convolutional layer is that it is easily scalable to a large number of inputs and extracts low-level feature representations to be fed into the rest of the network, which can consists of a regular hidden and classification layer, for example. A prime example of the application of a CNN to natural language processing is provided in Collobert et al. (2011), which has been discussed above.

1.5 The current project

In this project, an attempt will be made to create a neural network and semantic word vector-based system that can out-perform the current state-of-the-art in singleton detection, with the goal of boosting coreference resolution performance. Of all mention detection tasks, singleton detection is the broadest, most-challenging and likely most effective one when it comes to boosting coreference resolution performance.

Here, singleton detection will be performed as defined in de Marneffe et al. (2015). de Marneffe et al.'s task is recent, successful in its application, and shows state-of-the-art performance. For ease of comparison, and because they, too have a strong focus on evaluation in-line with coreference resolution, the same training and test data (OntoNotes v4.0, v5.0 Weischedel et al., 2011, 2013), evaluation metrics (CoNLL-scoring, Pradhan et al., 2011) and coreference resolution systems (Stanford and Berkeley systems Lee et al., 2013; Durrett and Klein, 2013) will be used. We choose the Stanford and Berkeley systems because they are both recent, high-performance systems, they are very different in nature (deterministic rule-based versus surface feature learning-based) and to facilitate comparison of results to de Marneffe et al.'s work.

Unlike things like syntactic structure or sentiment analysis, singleton detection, ultimately, depends on comparison rather than on composition. Sentiment analysis, for example, deals with how the sentiment of a sentence is made up of the sentiment values of its parts. Singletonhood, on the other hand, is determined by other mentions in the vicinity of the mention under consideration, and whether or not these are coreferential.

Because tasks like sentiment analysis consist of determining the value of the whole from the values of its parts, they are especially suited for solving using recursive architectures, such as the various recursive neural networks.

For singleton detection, this does not apply, as there are no singleton probabilities that can be assigned to the parts of a mention. Rather, singleton detection requires a system that can take in a set of information about the mention and its context, and learn based on that. This set has no inherent structure that can be exploited. As such, the multi-layer perceptron is the most suitable type of neural network for the task, and is the one that will be used in this project.

Regarding the other components of the singleton detection system, such as word vectors, some additional choices have to be made. Which alternatives are most suitable depends on the approach to the singleton detection task, i.e. the underlying idea of how singleton detection should be solved.

Ultimately, singletonhood can only be determined by performing full coreference resolution, and to use a coreference resolution system to do singleton detection would be to put the cart before the horse. As such, it is better to frame the singleton detection slightly differently: instead of trying to determine whether something is a singleton or not, the goal should be to predict the probability that a mention is singleton.

Although it does not fundamentally change the problem, it does help in figuring out how to approach the task, now we can ask the question: 'What influences singleton-probability?'. Here, a three-way distinction of what informs singleton-probability is proposed:

1. The nature of the mention itself. For example, a mention '*it*' is more likely to be singleton than a mention '*they*'.

2. The direct context, i.e. the words and phrases around the mention, within a certain span, such as in the same sentence. For example, '*it*' is more likely to be singleton in '*it* seems that' than in '*it* belongs to'.

3. Other mentions in the vicinity, for example other NPs in the same paragraph (or another span). For example, '*it*' is more likely to be singleton if the only other mention in the vicinity is '*the building*' than if that other mention is '*the women*'.

Most other approaches to mention detection have focussed on the first two types of context, although longer-distance head and string match features have also been used. Nevertheless, the incorporation of other NPs in the vicinity as features has the potential to really improve singleton detection, since it allows for comparisons to other higher-level units, which is what is normally done in the clustering part of coreference resolution.

For the representation of words, the GloVe semantic word vectors (Pennington et al., 2014) will be used. These pre-trained vectors are freely usable, and come in different dimensionalities (e.g. 50- and 300-dimensional vectors), so the effect of varying this can be explored. Additionally, these vectors have shown good performance when used for other NLP tasks. The vectors will not be adapted during training, both for reasons of complexity and for reasons mentioned in Section 1.4.3.

For the use of mentions/NPs in the input, representations will have to be generated from the words that make them up. The recursive auto-encoder is the best option for this, since it can be trained in an unsupervised manner and is especially suited for generating representations. Also, RAEs aim to keep as much information in the NP as possible, which should benefit classification performance.

With these components, a singleton detection system will be built and performance evaluated on the OntoNotes corpus and as part of the Berkeley and Stanford coreference resolution systems. The specifics of the datasets, system architecture, training and evaluation procedure are presented in the next section.

2 Methods

The applied aspect of a machine learning study like this consists of two main parts: the data used and the algorithms used to process that data. In this section, first, the used datasets will be discussed, and after that the preprocessing steps and algorithm that make up the singleton detection and evaluation system will be described.

2.1 Data

The choice for which set(s) of data to use for training and evaluation is important, because it is part of the definition of the task. It affects both the difficulty of the task and the comparability of the work to other work on the same topic. In addition, the quality of the dataset, such as the consistency of annotations, matters for the performance of a machine learning system. Usually, larger and higher-quality sets of training data make for better models.

Fortunately, in the field of coreference resolution, and by extension, singleton detection, the choice for a dataset is relatively simple. Both in terms of size, scope, quality of annotation, coverage of text types and wide usage (for comparability), the OntoNotes corpus is the best choice. This is the corpus used in the CoNLL 2011 and 2012 shared tasks (Pradhan et al., 2011, 2012, respectively), and is also used by de Marneffe et al. (2015), the most important work on singleton detection.

The CoNLL-2011 shared task made use of OntoNotes version 4.0 (Weischedel et al., 2011) and the CoNLL-2012 shared task used data from OntoNotes version 5.0 (Weischedel et al., 2013). As stated in the corpus documentation, the goal of the OntoNotes project is to: *"annotate a large corpus comprising various genres [..], with structural information [..] and shallow semantics"*.

The OntoNotes corpus comprises seven different genres of text: newswire, transcribed broadcast news, transcribed broadcast conversation, magazines, transcribed telephone conversations, web text (weblogs, newsgroups) and religious text. As is the case with many corpora, newswire-style text is dominant: the newswire and broadcast genres together make up over half of the words in the corpus.

The corpus is large, at least as far as coreference-annotated corpora go, as it contains approximately 1.5 million words of English text. All text is annotated with six layers of annotation: syntactic structure (including part-of-speech), predicate-argument structure, speaker annotation (if applicable), named entity tags, word sense information and coreference labelling. Since another goal of the project is to strive for 90% inter-annotator agreement, the annotation is of high quality. There is no difference between the shared task data and the OntoNotes corpus; in both shared tasks the whole corpus was used, with only a difference in formatting. As such, the terms OntoNotes version 4.0/5.0 and CoNLL-2011/2012 data will be used interchangeably from

Dataset	Documents	Tokens	Noun phrases	Coreferential mentions
2011 Training	1667	956K	312K	94,155
2011 Development	202	136K	44K	14,291
2011 Test	207	153K	49K	16,291
2012 Training	1933	1247K	405K	148,464
2012 Development	222	163K	53K	19,156
2012 Test	222	170K	54K	19,764

Table 4: The number of documents, tokens, noun phrases and coreferential mentions for the split CoNLL 2011 and 2012 data sets. Note that the number of noun phrases is not equal to the number of mentions, which is a more complex notion, cf. Section 2.1.1.

here on.

The OntoNotes dataset is divided into three subsets: a training set (80%), a development set (10%) and a test set (10%). This division was used in the CoNLL tasks and by de Marneffe et al., and will also be used here, for ease of comparison. Statistics on the size of the datasets and the number of NPs (as an approximation of the number of mentions) and coreferential mentions are displayed in Table 4.

Of the six annotation layers in the dataset, the coreference annotation layer is the most relevant one here and should thus be discussed in more detail. Coreference annotation, i.e. annotating which mentions in a text refer to the same entity, is not an unambiguous task. Therefore, as is the case with all manual annotation projects, a set of annotation guidelines is provided. These guidelines differ between corpora, and as a consequence, things that are coreferential in one corpus or project are not necessarily annotated as such in another. In the OntoNotes corpus, for example, only coreferential mentions are annotated and singleton mentions are not, while in the ACE corpus (Doddington et al., 2004), singleton mentions are annotated. How coreference is defined, exactly, can have a lot of influence on the effectiveness of a certain approach. Besides, annotation specifics influence scores on several of the established coreference resolution metrics, as shown by Recasens and Hovy (2010).

To cover the complete coreference guidelines here is unnecessary. Nevertheless, the most noteworthy aspects of coreference as defined within the OntoNotes corpus are worth mentioning and are as follows: generic mentions can be coreferential, but only with non-generic mentions, such as pronouns, as in Example 19, but not with other generic mentions (e.g. another mention of *a spokesman*). Subjects are not coreferential with their nominal predicates, as in Example 20 ($_x$ indicates non-coreferential). Temporal expressions are almost always considered candidates for coreference, even if the expressions are relative to the time of writing of the document (e.g. Example 21). Similar to the way predicates are treated, mentions that are part of appositive constructions are generally not considered coreferential. Rather, the whole appositive construction is considered as one mention, as illustrated by Example 22 (again, $_x$ indicates non-coreferential mentions).

(19) [A spokesman]$_1$ said [he]$_1$ …

(20) The South Korean government is establishing diplomatic relations with [Poland]$_1$. [Poland]$_1$ is [the second Communist nation]$_x$ to …

(21) [Three years from now]$_1$, there will be peace. [Then]$_1$ …

(22) … when [[the girl's brother]$_x$, [Abdullah Khaled al-Harbi (18 years old)]$_x$]$_1$, and [his]$_1$ friend …

2.1.1 Mention selection

Besides specifying which mentions should be considered coreferential, the OntoNotes guidelines also specify what should be considered as a mention in the first place. In the section on mentions, the guidelines identify 4 types of mentions: noun phrases, possessive nouns and pronouns, pre-modifiers and verbs.

All noun phrases are considered mentions, with the major exception that, in the case of nested NPs that have the same headword, only the NP with the largest span is considered a candidate for coreference. So, in Example 23, *the punishment of his killers* and *his killers* are mentions, but *the punishment* is not. In newswire texts in particular, this exception reduces the number of candidate mentions significantly, since long NPs containing a lot of information are common in this genre.

Possessives, such as *the girl's* and *his* in Example 24, are considered mentions without exception. In the case of pre-modifiers, only proper nouns, such as *Vienna* in Example 25 are considered mentions, but *arms-control* in *arms-control talks* is not, for example. For verbs, only those verbs that co-refer with another mention are considered as mentions themselves. However, verb mentions are not of interest, since verb and event coreference is outside the scope of the current work.

(23) … and demands [[the punishment] of [his killers]].

(24) … [the girl's] brother, Abdullah Khaled al-Harbi (18 years old), and [his] friend …

(25) Concerns about the pace of the [Vienna] talks …

Generally, mention selection is not very important for tasks like discourse-new classification and singleton detection. Usually, all mentions can be extracted from the corpus and used as training and evaluation data. However, since singleton mentions are not annotated in the OntoNotes corpus, one cannot simply extract all singleton mentions and all coreferent mentions and use those. Rather, the set of singleton mentions has to be inferred by collecting the set of all mentions and subtracting the set of coreferent mentions from that.

Of course, one approach is to simply extract all NPs (and possessives, pre-modifiers), subtract the coreferent mentions, predict singleton probabilities for all of them, and use only those mentions that are identified as such by the coreference resolution systems. The downside of this is that it pollutes the training data, which would then contain NPs like *the punishment*. These would be labelled as singleton, even though they might actually co-refer with a later mention; they are not annotated either way, so we cannot know for sure. This would hinder the system in actually learning a model that captures coreferentiality - causing it to classify a similar case, where the same NP is not nested and is thus a valid candidate for coreference, as singleton.

To summarize, including non-mention NPs and pre-modifiers in the training data would cause the system to learn aspects of both singleton detection and mention selection, where we only want it to learn singleton detection. Two possible solutions to this problem are discussed in the next section.

2.2 Pre-processing

To get from the CoNLL dataset to a set of singleton probabilities and coreference resolution performance scores, a series of steps is applied to the data, as illustrated in Figure 2. The first step in this pipeline is the pre-processing of the data to make it suitable as input to the recursive autoencoder, the next step in the pipeline.

The CoNLL data is a word-per-line column-separated formatting of OntoNotes, containing document, part and word indices and all annotated information for each word. The goal of pre-processing is to extract the set of mentions, along with all and only the relevant features of those mentions needed in the rest of the process.

Mention selection is the main component of this process, and it is complicated by the fact that the OntoNotes guidelines, the Berkeley and the Stanford coreference system all select their mention in a slightly different way.

One solution is to select all noun phrases, possessives and pre-modifiers, apply a head-finding algorithm to filter out the head-matching nested NPs, and apply a filter to the set of pre-modifiers that selects only the proper nouns. This would require an extra implementation step, but it would yield a set of mentions that is approximately equivalent to the set of corpus mentions.

However, these mentions would not necessarily match with the Berkeley and Stanford mentions, since both systems use their own head-finding algorithms, and apply a set of additional restrictions to their selection of mentions. In the case of the Berkeley system, this would cause some mentions to not be assigned a proper value for the features regarding singleton probability, and in the Stanford case, it would cause some mentions to not be considered in

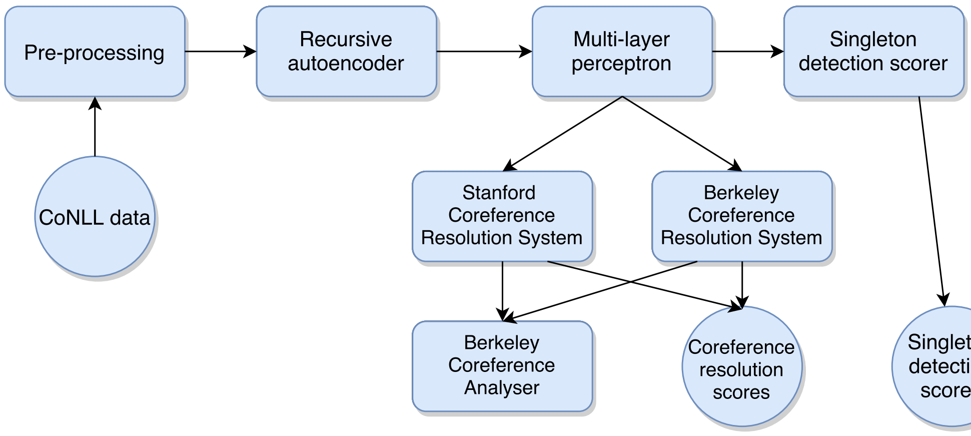
Pre-processing
Recursive autoencoder
Multi-layer perceptron
Singleton detection scorer
CoNLL data
Stanford Coreference Resolution System
Berkeley Coreference Resolution System
Berkeley Coreference Analyser
Coreference resolution scores
Singleton detection scores

Figure 2: A flowchart illustrating the singleton detection and evaluation system. Rectangles represent programs, circles represent datasets.

the pre-filtering step.

Therefore, a different approach is taken. Rather than trying to specify a set of mentions beforehand, we simply take the mentions as selected and used by the Stanford system, and use those as the base for the singleton detection data. Generally, the Stanford mentions match up well with the mentions used by the Berkeley system (Durrett and Klein state that their mention selection rules are based on those by Lee et al.), although not completely. We will return to this point in the discussion of the results, in section 3.3.

Then, for each of these mentions, a set of features needed further down in the pipeline is extracted. The mentions are identified by the following information: the document from which they are extracted, the index of the sentence in which they occur, and a begin and end index, marking the span of words that make up the mention.

As features, the words that make up the mention are collected, and n words before and after the mention are extracted. The words are not stored as words, but as indices, which link each word to its semantic word vector in the set of semantic word vectors. The word context of mentions at the beginning and end of a document is filled up with special padding tokens, if necessary. Words that do not have an entry in the vector dictionary are replaced by a special unknown word token, 'UNK', and are represented by a semantic word vector that is the average of all vectors in the vector dictionary. In the pre-processing step, a large context around each mention is extracted, but the size of the context actually used for the singleton detection can be specified in the model.

Finally, a label indicating whether or not the mention is coreferential is extracted by extracting the spans of all coreferent mentions and matching them to the set of all mentions.

2.3 Recursive Autoencoder

Before the multi-layer perceptron is trained to perform singleton detection and applied to the test data, the set of mentions is processed by a recursive autoencoder.

One reason for this extra processing step is that multi-layer perceptrons require a fixed-size input for all instances in the dataset. Naturally, one would like to (also) use the vector representations of the words that make up the mention, but since the mentions can contain almost any number of words, this would lead to great variations in the size of the input. To overcome this problem, a fixed-size vector representation for each mention is generated, from the vectors of the words it contains.

The second reason concerns one of the feature types used for the MLP: other mentions in the context of the mention that is under consideration. Using these mentions as features also requires them to have a fixed-size representation. Also, the intuition underlying the use of nearby mentions is that, the more similar a mention is to mentions in the neighbourhood, the more likely it is to be coreferential, and vice versa for dissimilar mentions. It is hypothesized here that this similarity can be captured better by the MLP when the context-mentions are represented by compact, fixed-size vector representations than if they were represented by variably sized sets of vectors representing individual words.

2.3.1 Algorithm

Recursive autoencoders (RAEs) are very well-suited for the generation of mention vectors, as they are able to generate representations of the same length as the individual input vectors, while aiming to preserve as much of the input information as possible. Naturally, we want to preserve as much of the information contained in the words that make up a mention, in order to maximize their value as features. Other advantages of RAEs are that they are relatively easy to implement, quick to train, and can be trained in an unsupervised manner.

As the name implies, autoencoders are used to find an encoding of the input. This encoding can be of any size, and the quality of the encoding is evaluated by performing a decoding step and comparing the reconstructed input to the original input; a good encoding will preserve most of the information in the input, allowing for a close reconstruction. The recursive aspect of a recursive autoencoder is quite straightforward: it simply means that the same autoencoder is applied repeatedly to a set of representations, until only

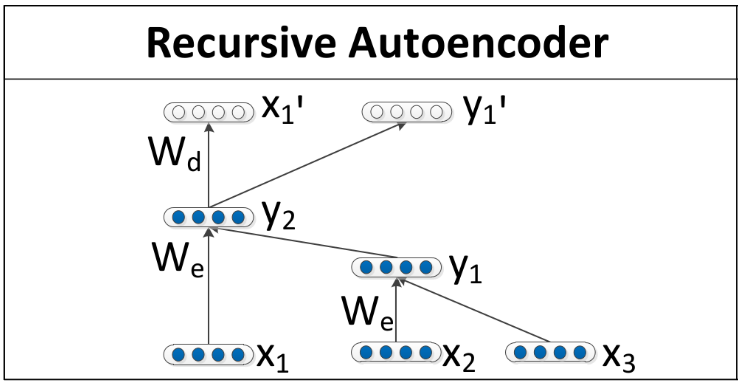

Figure 3: Illustration of the structure of a recursive autoencoder. The autoencoder consists of an encoding matrix, a decoding matrix, input vectors, hidden vectors, and reconstructed vectors. Diagram from Socher, Huang, Pennington, Ng and Manning (2011).

one representation is left. In the case of mention representations, this means that the autoencoder is applied to two n-dimensional word vectors, generates an n-dimensional encoding of these two vectors, which can then be combined with an additional word vector to generate a representation of three words together, and so on, until all word vectors have been integrated in the final n-dimensional mention vector.

The structure of the RAE set-up is illustrated in Figure 3. One 'run' or 'pass' of the RAE has the following phases: the encoding weight matrix, W_e, is applied to the 2 n-dimensional vectors, x_2 and x_3, mapping them to one n-dimensional vector, y_1 (Equation 1). The hidden vector, y_1 is then decoded by multiplying it by the decoding weight matrix, W_d, yielding the reconstructed input vectors x'_2 and x'_3 (Equation 2, not depicted in Figure 3). By calculating the difference between the reconstructed vectors and the original input vectors, we get the reconstruction error. In both the encoding and decoding step, after the multiplication by the weight matrix, a bias vector (b_e and b_d in Eq. 1, 2) is added to the resulting vector, after which a non-linear activation function is applied to the result ($f()$ in Eq. 1, 2).

$$y_1 = f(W_e * [x_2, x_3] + b_e) \tag{1}$$

$$[x'_2, x'_3] = f(W_d * y_1 + b_d) \tag{2}$$

The same process is applied to the intermediate representation y_1 and the input vector x_1 to get the mention vector y_2, spanning three words. After all input has been processed and reconstruction errors have been calculated for each set of input vectors, the backpropagation algorithm (cf. Section 2.4 is used to update the two weight matrices so that the reconstruction error is minimized. Since the error is defined solely by the reconstruction and the input, the data needs no annotation or labelling; the training is unsupervised.

2.3.2 Implementation

The implementation of the recursive autoencoder in the current project is similar to the basic RAE set-up just described. The weight vectors are initialized as small random values, and the bias vectors are initialized as zeroes. For the non-linear activation function, the sigmoid function is used ($f(a) = \frac{1}{1+e^{-a}}$). For the training, i.e. for the updating of the weights, stochastic gradient descent is used. This means that the weights are updated after each processed mention, instead of after the whole or part of the whole set. Another implementation decision is whether to train the autoencoder with so-called tied weights, which means that the decoding matrix is kept fixed as the transpose of the encoding matrix. We choose not to use tied weights here, because untied weights showed better generalization during pilot testing.

Another choice that has to be made is between the use of the internal syntax structure of the mention when applying the recursive autoencoder and the use of a general right- or left-branching binary tree structure. In the first case, the autoencoder is applied to those words that have the same parent node in the syntax tree, while in the latter case, the first two words are taken together, then the third is added to the intermediate representation of the first two, and so on.

Using the syntax structure is preferable when the intermediate representations are also of interest, as is the case in Socher, Huang et al. (2011), where both the complete and intermediate phrase representations were used as features for a paraphrase detection system. In this case, however, only representations covering the whole mention are relevant, so a left-branching binary tree structure is used, which has the advantage of being easier to implement.

The final choice to make in the implementation of the recursive autoencoder regards the error function, a measure of the difference between the original and reconstructed input vectors. A suitable error measure is to simply take the Euclidean distance between the input and reconstruction vectors, as in Equation 3.

For the recursive auto-encoder, however, this is not sufficient. When, for example, an intermediate representation that spans 7 words is combined with a word vector representing only 1 word, the regular error function would weigh the reconstruction of the single word vector as equally important to the reconstruction of the 7-word vector. Since this would cause the words combined later in the process to have a disproportionate influence on the final mention representation at the cost of words combined earlier, the regular error function needs to be adapted. This can be done by a simple weighting of each input vector, as proposed by Socher, Pennington, Huang, Ng and Manning

(2011). In this case, the reconstruction error of each vector is weighted by the number of words it represents, as in Equation 4 (n_2 and n_3 represent the number of words represented by x_2 and x_3, respectively).

$$Err = \frac{||x_2 - x'_2||^2 + ||x_3 - x'_3||^2}{2} \tag{3}$$

$$Err = \frac{n_2 * ||x_2 - x'_2||^2 + n_3 * ||x_3 - x'_3||^2}{n_2 + n_3} \tag{4}$$

2.4 Multi-layer perceptron

After application of the RAE, the input for the multi-layer perceptron consists of a set of mentions, their vector representations, and word vectors representing the words in the context of the mention. Together, these form the three types of features used for the singleton detection model: information from the mention itself, information from the direct context of the mention, and information from other mentions nearest to the mention under consideration. Depending on the size of the word vectors used, and the number of context words and mentions taken into consideration, this determines the size of the input. For example, using 50-dimensional vectors, the representation of the mention itself, and 5 context words (before and after) and 2 context mentions (before and after), this yields a $(1 + 5 + 5 + 2 + 2) * 50 = 750$ dimensional as input.

2.4.1 Algorithm

In addition to the input, the MLP consists, like the recursive autoencoder, of two weight matrices, two bias vectors and a single output, a probability in the $0 - 1$ range, which indicates the probability that the mention is coreferential. Figure 4 illustrates how these components fit together. For the example, we assume a number of 300 hidden nodes, but this number is free to vary during the parameter optimization process.

Training the multi-layer perceptron on a mention or a batch of mentions consists of two phases, or passes: the forward pass and the backward pass. In the forward pass, the values of the nodes in the hidden layer ($hid_1 - hid_{300}$) and the output node (out_1) are calculated. This is similar to the encoding phase of the autoencoder. The values of the input are multiplied by the first weight matrix (W_1) and the bias vector (b_1) is added. A non-linear activation function ($f()$, not the sigmoid function, but the tanh-function) is applied to the resulting values to get the output values of the hidden layer. The same process is then applied to these outputs, using W_2 and b_2 to get the value of the output node, which is a probability prediction. Equation 5 shows the forward pass in a single expression.

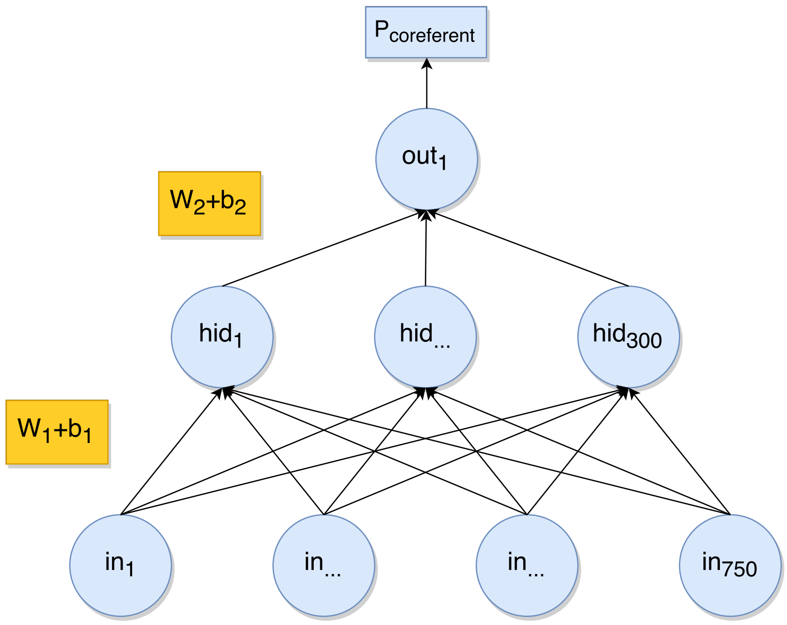

Figure 4: Illustration of the basic structure of the multi-layer perceptron used in this project.

$$P_{coref} = f(W_2 * f(W_1 * [in_1, .., in_{750}] + b_1) + b_2) \tag{5}$$

Training the multi-layer perceptron is a form of supervised learning, which means that the predicted probability has to be compared to the correct answer in order to get an error measure. All mentions are labelled as either singleton (0) or coreferent (1), and these are used as the 'ideal' probabilities for the MLP to learn. The goal is to maximize the predicted probabilities so that the predictions for singleton mentions are close to 0 and those for coreferential mentions are close to 1. Here, we use the negative log-likelihood as the error function (Equation 6, $P = P_coref$, $l = label$). This is a common error measure in machine learning, and minimizing the error serves to maximize the likelihood of the dataset given the model, that is, it gets the predictions as close to the correct labels as possible.

$$Err = -1 * (l * \log(P) + (1 - l) * \log(1 - P)) \tag{6}$$

After the forward pass has produced an output probability and an error value, the next phase in training is the backward pass. In the backward pass, the error value is propagated back through the network, using the aptly named backpropagation algorithm. The principle underlying backpropagation is to calculate the derivative of each weight with respect to the error function, which is essentially the same as calculating in what way and how much each weight is responsible for the found error. This derivative is used to adjust the weights in such a way that the error is minimized. The derivative is dependent on the type of error function, the set of weights that connect a node to the output, and the type of activation function used.

To give the complete derivation of the backpropagation algorithm here would be too much, but it is useful to look at the calculation of the adjustment for an example, e.g. a weight between an input node i and a certain hidden node h (Equation 7. To be able to calculate the effect this weight has on the output, we also need to know how node h affects the output node, which is given, for a certain node h and the output node o, in Equation 8. Both require the derivative of the error function, Err', given in Equation 9. The x in Equation 7 and 8 represents the input of the node in question. The α represents the learning rate, a macro parameter which determines the size of the change in weights.

$$\Delta w_{ih} = \Delta w_{ho} * w_{ho} * \tanh(a_h) * x_h * \alpha \tag{7}$$

$$\Delta w_{ho} = Err' * \tanh'(a_o) * x_o * \alpha \tag{8}$$

$$Err' = -\frac{label}{P_{coref}} + \frac{1 - label}{1 - P_{coref}} \tag{9}$$

The biases are not present in these equations, but they are trained in the same way. There is no essential difference between a weight and a bias; the bias can also be regarded as an additional weight, with a fixed input value of 1. In this way, the same training scheme can be applied to the biases as to the weights.

2.4.2 Implementation

As with the recursive autoencoder, the implementation of the multi-layer perceptron is quite straightforward. The weight matrices are initialized at small random values, the biases as zero vectors. The input order is randomized, to avoid spurious effects (caused by coherences within documents and subcorpora, for example). Like the autoencoder, the MLP is also trained using stochastic gradient descent, meaning that the weights are updated after every mention.

Other choices regard the learning rate, the number of training epochs, early stopping criteria for training, the size of the contexts used, the size of the hidden layer, and weight regularization. These things are all governed by macro-parameters, and their values and optimization is discussed in Section 3.1.

2.5 Integrating singleton detection into coreference resolution systems

Assuming the optimal singleton detection model has been found, trained and probabilities have been generated for all data sets, the remaining step is to integrate these probabilities into a coreference resolution system to assess their added value. Here, we use two coreference resolution systems for evaluation, the Berkeley system and the Stanford system. Using these two systems has the advantage of being able to evaluate the two main ways of integrating mention detection information into coreference resolution: as a pre-filtering step or as an integral part of the model.

2.5.1 The Stanford system

The Stanford coreference resolution system (Lee et al., 2013) is a deterministic rule-based system that works with a set of so-called sieves (sets of rules), which are designed by humans and based on linguistic knowledge. The sieves capture varying aspects of coreference, for example matching headwords or speaker-pronoun links. After the initial mention detection phase, the remaining mentions pass through each sieve and two mentions are linked together if the rule in the sieve applies. Each mention has values for a number of features, such as number and gender, and these are shared with all mentions in the same cluster.

The sieves are ranked so that the highest-precision sieve is applied first, so that the best sieves have the most influence. This has proven to be a successful approach, since the Stanford system was the highest scoring coreference resolution system of the CoNLL-2011 shared task participants, and a part of the hybrid rule-based and machine learning system that was the best in the CoNLL-2012 shared task (Fernandes et al., 2012).

In a later version of the Stanford system, a learned singleton detection system (Recasens et al., 2013) was incorporated in the system. This singleton detection model is a direct precursor to the de Marneffe et al. system. Given the sieve-based architecture of the Stanford system, the most natural way of integrating a singleton detection model is to simply add it as if it were another sieve, right after the initial mention selection sieve. This is what Recasens et al. and de Marneffe et al. did, and will also be done here.

Such a sieve can be applied in different ways. It can be applied very liberally, by simply excluding all mentions that are classified as singleton from being incorporated in any cluster. More conservatively, it can be applied to stop the linking of two mentions only when both mentions are classified as singletons.

Another restriction, found to be beneficial to performance by de Marneffe et al. (2015), is to apply the filter only to mentions that are not tagged as named entities. Finally, the probability threshold under which to classify mentions as singleton can be varied, increasing or decreasing the number of mentions that are filtered out. By lowering the threshold, recall is sacrificed for a gain in precision, and vice versa for an increase of the threshold value.

2.5.2 The Berkeley system

The Berkeley coreference resolution system (Durrett & Klein, 2013), on the other hand, is a learning-based model that relies on automatically generated surface-level features. It is currently the best-performing publicly available coreference resolution system for English. It works by assigning to each (pair of) mentions a number of features based on a set of pre-defined feature templates. These templates are relatively simple, being based on things like head words, mention length and distance between a pair of mentions.

The features derived from these templates are assigned values for each mention, and used in a mention-ranking approach. This means that all mentions are individually classified as either part of an existing cluster or starting a new cluster (i.e., singleton or antecedent), and each pair of mentions is classified as to whether or not the mentions corefer. Using a log-linear model for learning, Durrett and Klein report CoNLL metric F-scores of approximately 3.5 percentage points above those of the Stanford system on the CoNLL-2011 development and test sets.

Because the Berkeley system is learning-based and works with features, it allows for the integration of singleton probabilities not as a pre-filter, but as features, incorporated them in the resolution model.

The most basic way to incorporate singleton probabilities is to add a feature containing the predicted probability for each individual mention. Another option is to add binary features to each mention based on a probability threshold, for example an 'isSingleton'-feature for probabilities under 0.2 or 0.3. This can also be done for pairs, where a feature is added if both mentions have singleton probabilities below or above a certain threshold.

A final possible variation is to restrict the use of predicted probabilities to the set of test data. Instead of using predicted probabilities for the training data, coreference labels are used as 'probabilities' for the training mentions. This has the advantage of using 100% accurate information for the training of the model, instead of using predicted probabilities, which contain errors. On the other hand, it has the disadvantage that it creates a large discrepancy between the distribution of feature values in the training data and the test data, which could hurt generalization performance. This has not been tested before, so it is not yet known whether the advantages outweigh the disadvantages.

No choice is made beforehand with regard to the integration of singleton probabilities in either the Berkeley or the Stanford system. Rather, the effect of these various possibilities will be evaluated and results reported in Section 3.2.

3 Evaluation & Results

Before the training and testing of the singleton detection model started, the two datasets, CoNLL-2011 and CoNLL-2012, were preprocessed using the method described in Section 2.2. The mentions selected by the Stanford coreference resolution system were used. A word context of 20 words before and after the mention was extracted, to allow for multi-layer perceptron training with up to 20 context words.

The next step of preprocessing was the application of the recursive autoencoder to generate word vector-length representations for each mention. During pilot-testing using a small sample of the corpus, the optimal learning rate was found to be 0.005, and the same rate was used for the rest of the corpus. Training was done using the training set of the CoNLL-2011 and 2012 data, respectively, and the trained RAE model was used to generate representations for the development and test sets.

The duration of training was determined by validation on the development set. After each epoch (one pass over the complete dataset) of training, the mean error on the development dataset was calculated. Training was stopped when the lowest mean validation error on the development data was not achieved in the last 25% of epochs, i.e. training had continued for 25% of total runs past the best model. The resulting representations were used throughout all subsequent training and evaluation.

3.1 Model optimization

The performance of the singleton detection model was evaluated on the CoNLL-2012 development set, using the CoNLL-2012 training set as training data. Only the final, optimized model was tested on the test set. The stopping criterion for training was similar to that for the recursive autoencoder: after each epoch, performance is validated on the development set by calculating the accuracy $(1 - Err)$. If the epoch with the highest development set accuracy was not in the last 25% of epochs, training was stopped. This saves the model from overfitting on the training data, while still training for a number of additional epochs large enough to avoid getting stuck in sub-optimal solutions.

The learning rate was kept fixed at the commonly used value of 0.001. During preliminary testing, no reason was found to change the learning rate, and keeping it fixed helps in isolating the influence of other parameters. In the rest of this section, multiple values for the rest of the parameters will be considered and an attempt is made to find the optimal model. Of course, due to the number of interactions between parameters and the non-deterministic nature of the model, finding the true optimum is not possible, so the aim is to explore the parameter space to a reasonable extent.

Unless indicated otherwise, the following unoptimized default parameter values were used: 50-dimensional word vectors, 150 hidden nodes, 5 context words (before and after), 2 context NPs (before and after), a 0.5 threshold for classifying mentions as singleton or coreferent, and the learning rate and training stopping criterion described above. The maximum number of training epochs was set to 200.

To put the performance of the singleton model in perspective, a competitive baseline was established. For the baseline, each single-word mention tagged as a pronoun or possessive is marked as coreferent, while all other mentions are labelled singleton. This yields a reasonably high baseline accuracy of 68.19%.

3.1.1 Hidden layer size

The first parameter to be taken into consideration is the number of hidden nodes. The optimal size for the hidden layer depends on the size and complexity of the input, and is hard to determine by itself. It is reasonable to expect that the hidden layer size has to scale linearly with the size of the input. For example, using 300-dimensional word vectors instead of 50-dimensional vectors, the size of the input sextuples, and the size of the hidden layer needs to be increased accordingly. Here, we attempt to find a good hidden layer size for a fixed-size input, and in subsequent tests, scale the hidden layer to the size of the input, keeping the input-to-hidden-layer-proportion fixed. The effect of different numbers of hidden nodes on singleton detection performance is shown in Table 5.

In addition to overall accuracy, performance is measured by the precision, recall and F1-score (harmonic mean of precision and recall), for both singleton and coreferent mentions. In the case of singletons, precision indicates the proportion of true singletons in the set of all mentions predicted to be singleton, where recall indicates the proportion of singleton mentions that were correctly predicted to be singleton. In other words, precision indicates how trustworthy the model's predictions are, while recall indicates how good the model is at identifying a certain type of mention.

As can be seen in Table 5, the hidden layer size only has a small effect on the overall model accuracy. In order to know whether any of these models is a significant improvement over the default hidden layer size of 150, pair-wise approximate randomization tests are performed (Yeh, 2000). These shows that not one of the models is significantly different from the 150 hidden nodes model with regards to accuracy, $p > 0.05$. As such, we choose an input/hidden layer proportion of 5:1 for the rest of the optimization process.

3.1.2 Number of context words

The next parameter value to explore is the one governing the number of context tokens that is taken into the input (Table 6). The results show that the

Hidden layer size	Overall Accuracy	Singleton Precision	Singleton Recall	Singleton F1-score	Coreferent Precision	Coreferent Recall	Coreferent F1-score
Baseline	68.19	66.90	**87.21**	75.72	71.91	43.13	53.92
50	76.88	75.90	**86.93**	**81.04**	**78.70**	63.62	70.36
100	77.07	76.88	85.34	80.89	77.41	66.18	71.35
150 (default)	77.12	77.28	84.62	80.79	76.84	67.23	71.71
300	77.08	76.54	86.08	81.03	78.05	65.24	71.07
600	**77.39**	77.56	84.74	80.99	77.10	67.70	72.09
800	77.24	**78.59**	82.42	80.46	75.25	**70.41**	**72.75**

Table 5: Singleton detection performance on the CoNLL-2012 development set, using the default multi-layer perceptron model with varying hidden layer sizes. Models that differ significantly in accuracy from the default model are marked with a *.

Context tokens	Overall Accuracy	Singleton Precision	Singleton Recall	Singleton F1-score	Coreferent Precision	Coreferent Recall	Coreferent F1-score
Baseline	68.19	66.90	87.21	75.72	71.91	43.13	53.92
0	75.44*	74.82	85.60	79.85	76.58	62.04	68.55
1	**77.64***	77.49	85.51	**81.30**	77.89	67.27	**72.19**
2	77.50*	77.32	85.51	81.21	77.81	66.94	71.97
3	77.11	76.44	86.35	81.09	78.30	64.93	70.99
4	77.01	76.63	85.70	80.91	77.68	65.55	71.10
5 (def.)	77.12	77.28	84.62	80.79	76.84	67.23	71.71
10	76.33*	**77.55**	82.14	79.78	74.48	**68.67**	71.46
15	76.04*	75.22	86.27	80.37	77.57	62.56	69.26
20	75.26*	72.71	**90.42**	80.61	**81.41**	55.29	65.86

Table 6: Singleton detection performance on the CoNLL-2012 development set, using the default multi-layer perceptron model, with a 5:1 hidden layer size. Varying numbers of context tokens are used as input. Models that differ significantly in accuracy from the default model are marked with a *.

Context mentions	Overall Accuracy	Singleton Precision	Singleton Recall	Singleton F1-score	Coreferent Precision	Coreferent Recall	Coreferent F1-score
Baseline	68.19	66.90	**87.21**	75.72	71.91	43.13	53.92
0	77.20*	76.72	85.97	81.09	78.03	65.63	71.30
1	77.38	77.35	85.14	81.06	77.43	67.16	71.93
2 (def.)	**77.64**	77.49	85.51	**81.30**	77.89	67.27	72.19
3	76.99*	76.85	85.19	80.81	77.23	66.19	71.28
4	77.49	**77.73**	84.65	81.05	77.09	**68.04**	**72.29**
5	77.27*	77.24	85.09	80.98	77.32	66.95	71.76
6	77.33*	76.83	**86.09**	81.20	**78.21**	65.78	71.46

Table 7: Singleton detection performance on the CoNLL-2012 development set, using the default multi-layer perceptron model, with a 5:1 hidden layer size. 1 context word before and after the mention are added to the input. The number of context noun phrases in the input is varied. Models that differ significantly in accuracy from the default model are marked with a *.

number of context words in the input has only limited effect on classification accuracy, which is somewhat unexpected.

Even more surprisingly, the best-performing model is the one that takes only 1 context word before and after the mention into account. Nevertheless, testing clearly shows that a lower number of context words increases classification accuracy, with no context at all being the exception to this trend. Both the models with 1 and 2 context words are significantly better than the default model. For subsequent evaluation, a 1-word context will be used, since it yields a higher accuracy than the 2-word context model.

3.1.3 Number of context mentions

The next parameter value to explore is the one governing the number of context mentions added to the input. Results are shown in Table 7.

Varying the number of neighbouring mentions in the input does not lead to an improved model. Apparently, the default value of 2 mentions before and after the current mention is already the optimal value, since all other models yield lower accuracies. As such, the number of context mentions in subsequent testing is kept at the default value of 2.

3.1.4 Word vector sets

Finally, the effect of different sets of pre-trained vectors on singleton detection performance is evaluated. Vector files of varying dimensionality and training corpus size are tested, e.g. 50D/6B is a set of 50-dimensional vectors, trained on a 6B word training corpus. Results are shown in Table 8.

Vector file	Overall	Singleton			Coreferent		
	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
Baseline	68.19	66.90	**87.21**	75.72	71.91	43.13	53.92
50D/6B (def.)	77.64	77.49	85.51	81.30	77.89	67.27	72.19
100D/6B	78.52*	79.41	84.00	81.64	77.18	71.30	74.13
200D/6B	79.01*	78.79	**86.30**	82.38	**79.36**	69.40	74.05
300D/6B	79.08*	79.81	84.61	82.14	77.97	71.79	74.76
300D/42B	**79.57***	79.77	85.83	**82.69**	79.25	71.23	**75.08**
300D/840B	79.27*	**79.98**	84.75	82.30	78.19	**72.05**	75.00

Table 8: Singleton detection performance on the CoNLL-2012 development set, using the default multi-layer perceptron model, with a 5:1 hidden layer size. 1 context word before and after the mention are added to the input. Vector files varying in dimensionality and size of training corpus are evaluated. Models that differ significantly in accuracy from the default model are marked with a *.

Test set	Overall	Singleton			Coreferent		
	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
2011-dev	76.37	75.72	90.32	82.38	78.11	54.40	64.13
2011-test	77.47	77.26	88.42	82.46	77.91	61.10	68.49
2012-dev	79.57	79.77	85.83	82.69	79.25	71.23	75.08
2012-test	80.08	81.57	83.78	82.66	78.01	75.25	76.60
2012-dev (dM15)	79.0	81.1	80.8	80.9	76.4	76.6	76.5

Table 9: Singleton detection performance on the CoNLL-2011 and 2012 development and test sets, using the best-scoring model on the 2012 development set. The CoNLL-2012 training set was used as training data. 'dM15' marks the results by de Marneffe, Recasens and Potts (2015)

Compared to the other parameters, the vector file has the largest effect on singleton detection performance. All models are significantly better than the default model, which uses a set of 50-dimensional vectors. Higher-dimensionality vectors perform better than lower-dimensional vectors, and vectors trained on larger corpora perform better than those trained on smaller corpora. The 300-dimensional vectors trained on a 42 billion word corpus yield the highest accuracy, and the model using these vectors will be used as the final model. The final model is used for testing on the other datasets and for the integration with coreference resolution systems.

3.1.5 Final model

Since all singleton detection models were both validated and evaluated on the 2012 development set, model training was stopped when the optimal accuracy on the development set was reached. Therefore, it might be the case that the results may not generalize well to other datasets. To test whether the current training set-up leads to good general models, the best model is applied to the other datasets. The best model is the one using the 300-dimensional vectors, trained on a 42B word corpus, with 1 context word, 2 context mentions and a 5:1 hidden layer size.

The results of evaluation on the CoNLL-2011 development and test set, and the CoNLL-2012 test set are reported in Table 9. Performance generally seems to hold up well across test sets. Performance on the 2012 test set is slightly higher even than on the development set, showing that the model generalizes well within the CoNLL-2012 corpus. However, results on the 2011 sets are clearly lower. In addition to being lower in accuracy, there is also a clear difference in the other metrics. The 2011 sets show higher recall and lower precision for singletons, and approximately equal precision and lower recall for coreferents.

A possible explanation for this is that the model is over-fitting on the singleton/coreferent proportion in the validation dataset. If the 2012 development set has a higher percentage of singleton mentions than the 2011 data sets, we would expect the model to label too many mentions in the 2011 sets as singleton, leading to higher recall, but lower precision. Unfortunately, this cannot be the case, as the 2012 development set has 56.85% singletons, while the 2011 development and test sets contain 61.16% and 59.92% singletons, respectively. So, actually, one would expect scores to change in the opposite direction. This means that the model labels a much higher percentage of mentions as singleton in the 2011 data sets than in the 2012 sets, the reasons for which are not immediately clear.

Table 9 also shows a comparison to de Marneffe et al.'s singleton detection system. The accuracy of our system is slightly higher, by 0.6 percentage points. We can only compare scores on the 2012 development set, since De Marneffe et al. do not report scores on other data sets.

Although accuracy is approximately equal, the systems differ more on the other metrics. The current system over-classifies things as singleton, which leads to higher recall and lower precision for singletons, and vice versa for coreferent mentions. De Marneffe et al., on the other hand, report almost equal precision and recall in both cases, which indicates that their system better captures the singleton/coreferent ratio in the data.

A similarity between the two systems is that they both score higher on singleton detection than on coreferent detection. This should not be taken to mean that singleton detection is easier than coreferent detection. Rather, it is a reflection of the fact that both systems optimize overall accuracy, in which case it is beneficial to score better on singletons, which occur more often, instead of doing equally well on singletons and coreferents.

3.2 Coreference resolution results

In addition to optimizing the singleton detection model, possible gains are also to be made by better integration of singleton detection in coreference resolution systems. In this section, the effect of adding the singleton predictions to the Stanford and Berkeley systems in several ways will be presented.

Tables 10 and 11 show the performance of the Stanford coreference resolution system on the CoNLL 2012 development set, for several ways of integration with the singleton probabilities. There are two baseline values: 'No Probabilities', which is the system without any singleton filtering, and 'Recasens et al. (2013)', which is the Stanford system in its current version, which has the singleton detection model from Recasens et al. integrated.

The other models are named as follows: 'All' indicates filtering is applied to all mentions, 'NonNE' indicates filtering is applied to non-NE mentions only. 'Mentions' indicates that individual mentions are prohibited from linking, while 'Pairs' indicates that only links between pairs are prevented, when both are identified as singleton. '0.5/0.15' indicates the threshold under which mentions are classified as singleton.

The threshold value of 0.15 is chosen so that the singleton classification has a precision of approximately 90%. This reduces the recall of the model, i.e. it reduces the number of mentions filtered from the resolution system, but makes sure that the filtered mentions are more likely to actually be singleton. The effect of varying the threshold on classification precision and recall was determined for the best model on the 2012 development set, and is shown in Figures 5 and 6. This was used to determine the threshold values for both singleton and coreferent mentions for which classification reaches 90% precision.

The performance scores of the varying filters follow the expected pattern. For more widely-applied filters (e.g. 'All-Mentions-0.5), precision goes up, but recall drops. For more selective filters, the drop in recall is smaller, but so is the gain in precision. The best filtering method is therefore the one that gains the most precision, at the relatively smallest cost in recall. Here, this is 'All-Pairs-0.15', which yields the highest CoNLL-F1 score and also the highest F1-score for 3 of the 5 metrics. Surprisingly, the restriction of filtering to non-named entities only is not beneficial, which contrasts with the findings of de Marneffe et al. (2015).

Since the effect of the singleton filtering is only small when compared to the baseline, it is not directly clear whether the differences are significant. To test for this, a paired-bootstrap re-sampling test (Koehn, 2004) over documents is used to compare the scores for each metric of each filtering model to the baseline. During re-sampling, a test set the size of the development set was sampled 10000 times. The same method is applied to significance testing for the results in Tables 12 and 13.

The results reported here for the 'No probabilities'-baseline differ from the scores reported for the same system in de Marneffe et al. (2015), because they use a different, older version of the CoNLL-scorer. Unfortunately, this also prevents direct comparison between the effect of their singleton detection system and the one presented here. Since de Marneffe et al.'s system is not publicly available, their results cannot be reproduced using the same scorer. The integrated singleton detection by Recasens et al. provides a good proxy for comparison, since it is very similar to the de Marneffe et al. system, although less advanced. Another option would be to use the old scorer, but using a faulty scorer distorts the actual performance of all systems, so this is not a viable option.

Ultimately, the 'All-pairs-0.15' filter yields only a small improvement, of 0.7 percentage points in CoNLL F1-score. This is more than the Recasens et al. filter, but only by a small margin. It does out-perform the Recasens et al. filter on almost all metrics, indicating that it improves performance overall, with both a larger precision gain and a smaller recall loss. De Marneffe et al. report a similar gain on the 2012 development set, of 0.5 percentage points.

	CoNLL	MUC			B^3		
Model	F	P	R	F	P	R	F
No Probabilities	56.44	64.74	**64.40**	64.57	57.36	**51.29**	54.15
Recasens et al. (2013)	56.90*	65.86*	64.09*	64.96*	58.61*	50.91*	54.49*
All-Mentions-0.5	50.41*	**74.32***	52.39*	61.46*	**66.68***	37.54*	48.04*
All-Mentions-0.15	56.73	69.41*	62.12*	**65.57**	62.07*	48.24*	54.29
All-Pairs-0.5	55.46*	68.87*	60.25*	64.27*	61.66*	46.42*	52.97*
All-Pairs-0.15	**57.17***	66.32*	64.14*	65.21*	59.12*	50.93*	**54.72***
NonNE-Mentions-0.5	53.64*	70.96*	57.54*	63.55*	63.02*	43.26*	51.30*
NonNE-Mentions-0.15	56.71	67.72*	63.05*	65.30	60.22*	49.40*	54.28
NonNE-Pairs-0.5	55.96*	67.38*	61.73*	64.43*	60.09*	48.20*	53.50*
NonNE-Pairs-0.15	56.92*	65.68*	**64.22***	64.94*	58.46*	**51.06***	54.51*

Table 10: Coreference resolution performance of the Stanford system on the CoNLL 2012 development set. All evaluations run with the Stanford CoreNLP suite, version 3.5.0 and version 8.0.1 of the CoNLL scorer. POS, lemma, NE and parsing annotation was used, in addition to a dataset containing number and gender information. Scores that are significantly different ($p < 0.05$) from the 'No Probabilities'-baseline are marked with a *. Best scores for each metric are in bold.

The performance of the singleton detection model when integrated with the Berkeley coreference system is presented in Tables 12 and 13. The baseline model, without singleton detection, is 'No Sing. Det.'. The singleton

CEAF-m			CEAF-e			BLANC		
P	R	F	P	R	F	P	R	F
59.92	57.94	58.92	47.09	**54.67**	50.60	58.22	**54.23**	55.64
59.17*	59.62*	59.39*	48.53*	54.27*	51.24*	59.44*	53.74*	56.03*
66.06*	45.25*	53.71*	**54.46***	33.83*	41.73*	**69.49***	41.00*	51.37*
62.07*	56.40*	59.10	51.59*	49.15*	50.34	63.41*	51.17*	**56.55**
61.78*	54.92*	58.15*	50.85*	47.53*	49.14*	62.95*	49.39*	55.28*
59.66*	59.60*	**59.63***	49.20*	54.20*	**51.58***	59.99*	53.89*	56.40*
63.96*	51.07*	56.79*	53.40*	40.51*	46.07*	65.40*	45.65*	53.73*
60.86*	57.65*	59.21	50.57*	50.56*	50.56	61.46*	52.20*	56.31
60.52*	56.75*	58.58*	49.77*	50.11*	49.94*	61.19*	50.89*	55.41*
59.04*	**59.75***	59.39*	48.51*	**54.45***	51.31*	59.28*	**54.01***	56.10*

Table 11: Continuation of Table 10.

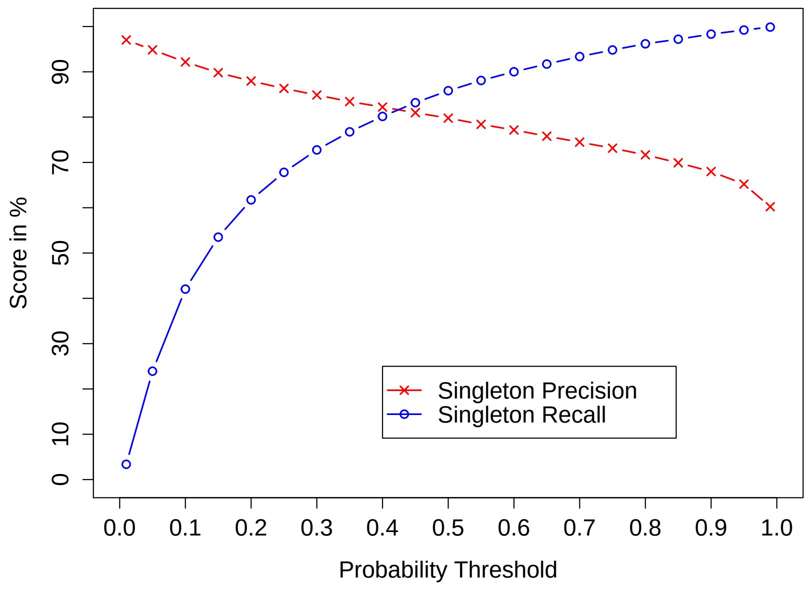

Figure 5: The effect of varying the classification threshold probability on precision and recall of singleton classification.

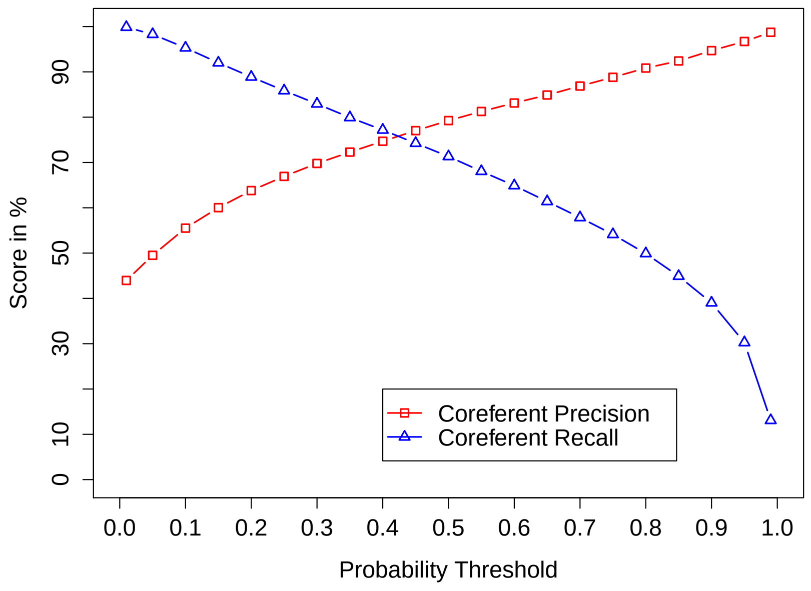

Figure 6: The effect of varying the classification threshold probability on precision and recall of coreferent classification.

probabilities were integrated in five different ways: 'Prob' indicates each mention was assigned it's predicted probability as a feature. 'Mentions' indicates each mention was assigned a boolean feature indicating whether it was singleton ($P < 0.15$) and a boolean feature indicating whether it was coreferent ($P > 0.8$). 'Pairs' indicates the same as 'Mentions', but for pairs of mentions, where both have $P < 0.15$ or $P > 0.8$. 'Both' indicates that both 'Mentions'- and 'Pairs'-features are added. 'Labels' indicates that coreference labels are used for the feature values of the training data, instead of predicted probabilities.

Overall, the pattern is similar to that in the results for the Stanford system, but the differences are smaller. The 'Prob-Both-Labels' is an outlier and performs a lot worse than any of the other systems. The 'Prob-Pairs' and 'Prob-Both' models are almost identical in their performance, and together are the two best performing methods of integration. However, they still yield only a 0.3 percentage point increase in CoNLL-F1 score over the baseline, and this is not a significant difference. Both models show improvements in precision, at only a small recall cost, but the improvements are too small to have any real effect.

The 'Prob-Both-Labels' model does yield larger increases in precision, but at a very large recall cost, so it is worse overall. This is likely because the use of training labels activates the boolean features for all mentions in the training data, which leads to overzealous filtering of mentions in the test data

Model	CoNLL F	MUC P	MUC R	MUC F	B^3 P	B^3 R	B^3 F
No Sing. Det.	61.71	73.44	67.68	70.44	63.14	55.55	59.10
Prob	61.83	73.39*	**67.75***	70.46	63.10*	**55.76***	59.20
Prob-Mentions	61.81	73.97*	67.33	70.49	64.08*	55.23	59.32
Prob-Pairs	**62.02**	74.03*	67.55	**70.64**	64.06*	55.62	59.54
Prob-Both	**62.02**	74.08*	67.51	**70.64**	64.15	55.59	**59.56**
Prob-Both-Labels	57.28*	**74.98**	60.70*	67.09*	**64.35**	47.55*	54.69*

Table 12: Coreference resolution performance of the Berkeley system on the CoNLL 2012 development set. All evaluations run with version 1.1 of the Berkeley Coreference Resolution System and version 8.0.1 of the CoNLL scorer. As a baseline system, the system using the 'FINAL' feature set was used. Scores that are significantly different ($p < 0.05$) from the 'No Probabilities'-baseline are marked with a *. Best scores for each metric are in bold.

that also have these boolean features (i.e. also have $P < 0.15$ or $P > 0.8$).

Comparing the results to those of de Marneffe et al. directly is not possible here, either. Still, we see a large discrepancy. Where they report a smaller effect of singleton detection on the Stanford system, they report a larger increase of almost 2 percentage points for the Berkeley system, on the 2012 development set. As such, we would also expect our singleton detection model to have a larger influence on the Berkeley than on the Stanford system. A direct explanation for this difference is not at hand, other than there is a possible qualitative difference in the predicted probabilities between our system and the one by de Marneffe et al.

One possible explanation for the poor performance with the Berkeley system is that the singleton detection system predicts probabilities for the mentions as they are identified by the Stanford system. These mentions are not the same as the ones used by the Berkeley system, which causes a number of the mentions in the Berkeley system to not have a predicted probability. In these cases, the average predicted probability is imputed as a value, which cannot be beneficial to performance. However, the same mentions were used by de Marneffe et al., and they did not find the same effect, so this cannot be a full explanation.

3.3 Coreference resolution analysis

To get a better insight in whether the singleton detection helped coreference resolution in the way it is expected to, it is useful to look at the types of errors made by both systems, with and without singleton detection. It can also help in figuring out the reason for the performance discrepancy between

CEAF-m			CEAF-e			BLANC		
P	R	F	P	R	F	P	R	F
65.63	60.95	63.20	57.00	54.21	55.57	62.88	58.08	60.34
64.24*	**61.74***	62.96	57.17*	**54.56***	55.84	61.03*	**59.70***	60.33
64.97*	61.23	63.04	57.18*	54.14	55.62	62.62*	58.59	60.51
65.28	61.69	**63.44**	**57.59**	54.26	**55.87**	63.09	59.00	60.69
65.27	61.59	63.38	57.53	54.25	55.84	62.99	58.90	**60.86**
65.71	53.50*	58.98*	55.21	45.79*	50.06*	**63.88**	50.62*	56.11*

Table 13: Continuation of Table 12.

the Stanford and Berkeley systems.

In order to do this, the Berkeley Coreference Analyser is used (Kummerfeld & Klein, 2013). This tool classifies coreference resolution errors as one of 7 types, counts them, and marks them in text for manual inspection. The 7 types of errors are (examples from Kummerfeld and Klein, 2013):

1. **Span Error** - the span of the mention is not predicted correctly
   a. Has: *Soviet leader [Gorbachev]*
   b. Should have: *[Soviet leader Gorbachev]*
2. **Conflated Entities** - two separate entities are clustered together
   a. Has: *[Mohammed Rashid]*$_1$, *[the Rashid case]*$_1$, *[the case]*$_1$, *[Rashid]*$_1$
   b. Should have: *[Mohammed Rashid]*$_1$, *[the Rashid case]*$_2$, *[the case]*$_2$, *[Rashid]*$_1$
3. **Extra Mention** - a singleton mention is added to a cluster
   a. Has: *[her story]*$_1$, *[this]*$_1$, *[it]*$_1$
   b. Should have: *[her story]*$_1$, *[this]*$_1$, *it*
4. **Extra Entity** - a cluster that does not exist is predicted
   a. Has: *[human rights]*$_1$, *[Human Rights]*$_1$
   b. Should have: *human rights, Human Rights* (no cluster)
5. **Divided Entity** - A single entity is spread across two clusters
   a. Has: *[Iraq]*$_1$, *[this nation]*$_2$, *[this nation]*$_2$, *[its]*$_1$
   b. Should have: *[Iraq]*$_1$, *[this nation]*$_1$, *[this nation]*$_1$, *[its]*$_1$
6. **Missing Mention** - a mention is not clustered, even though it should be

	Without SD	With SD	Difference
Span Error	434	428	-6
Conflated Entities	2103	2068	-35
Extra Mention	746	725	-21
Extra Entity	1295	1059	-236
Divided Entity	2546	2538	-8
Missing Mention	1190	1188	-2
Missing Entity	909	948	+39

Table 14: Error counts as produced by the Berkeley Coreference Analyser, for the Stanford coreference resolution system with and without singleton detection.

a. Has: *[the Arab region]$_1$, [the region]$_1$, it*

b. Should have: *[the Arab region]$_1$, [the region]$_1$, [it]$_1$*

7. **Missing Entity** - a cluster is not created, even though it should be

a. Has: *the pills, the tranquilizing pills*

b. Should have: *[the pills]$_1$, [the tranquilizing pills]$_1$*

The coreference analysing tool was applied to the baseline Stanford and Berkeley models, and to the best singleton detection models, 'All-Pairs-0.15' and 'Prob-Both', respectively. The errors for the Stanford systems are presented in Table 14, and the errors for the Berkeley systems are presented in Table 15.

Table 14 shows that singleton detection affects the coreference resolution largely as expected. Because a number of singleton mentions is filtered out beforehand, the number of conflated entities, extra mentions and extra entities drops, while the number of missing entities rises. The decrease in extra mentions is the most direct result of singleton filtering; as there are less mentions to consider, less mentions are erroneously added to clusters.

The fact that most of the overall improvement comes from the decrease in extra entities is caused by singleton mentions being removed from the output. When singleton mentions are filtered out, less singleton mentions are erroneously linked to other singleton mentions. These are then filtered out, which leads to a decrease in clusters that do not exist. This decrease also likely the cause of the decrease in conflated entities; if there are less entities, they are less easily conflated.

For singleton filtering to have a positive effect, it has to keep the increase in errors as low as possible. Missing mentions and missing entities are the error types most likely to see an increase due to singleton filtering. When mentions are filtered out, they are not added to entities, leading to clusters missing a mention, or clusters not being formed at all.

	Without SD	With SD	Difference
Span Error	391	367	-24
Conflated Entities	1706	1661	-45
Extra Mention	646	628	-18
Extra Entity	740	693	-47
Divided Entity	2057	2044	-13
Missing Mention	1019	1043	+24
Missing Entity	1016	1065	+49

Table 15: Error counts as produced by the Berkeley Coreference Analyser, for the Berkeley coreference resolution system with and without singleton detection.

Here, the increase in missing entities is relatively small, compared to the decrease in extra entities, and the number of missing mentions does not increase at all. This is reflective of the fact that the filtering model only filtered mentions with a low predicted probability ($P < 0.15$), making the chance that filtered mentions were actually coreferential quite low, which in turn keeps the error counts down.

De Marneffe et al. carried out a similar analysis of their errors, and the effect of their singleton detection system on the counts for various error types is almost identical to that just discussed.

Looking at the results for the Berkeley systems, we see that the error counts for missing entity and missing mention are slightly higher, but not much more than for the Stanford systems. Unfortunately, the decrease in the other errors is a lot smaller, which explains the lack of a significant increase in overall coreference resolution performance. Here, the increase in missing entities is approximately as large as the decrease in extra entities, where the decrease was 6 times as large for the Stanford case. Apart from that, the decrease in conflated entities and extra mentions is comparable.

Clearly, the main difference between the effect on the Stanford system and the Berkeley system is the lack of effect on extra entity-errors. However, since the integration with the Berkeley system is not a simple hard filter, the reasons behind this are not obvious.

Looking at de Marneffe et al.'s analysis for the Berkeley system, we see that they do report a similar decrease in extra entities, but almost no increase in missing mentions. In addition, they report a large decrease in divided entities, which we do not see here, and a larger effect on extra and missing mentions, which we do not see here either.

That de Marneffe et al.'s overall positive effect does not come from the extra entity type errors either is perhaps not surprising, since the Berkeley system already has a relatively low proportion of these errors, when compared to the Stanford system.

Clearly, the overall positive effect is due to the decrease in divided entity type errors. Somehow, the Berkeley system manages to utilize their singleton probabilities in such a way that it decreases divided entity errors, but it cannot do the same for our probabilities. What the reason for this is, is unclear.

4 Discussion

In this project, a singleton detection system was developed that works with semantic word vectors, a recursive autoencoder and a multi-layer perceptron in order to perform singleton detection on the CoNLL-2011 and 2012 datasets, and evaluate effectiveness in-line with the Berkeley and Stanford coreference resolution systems. The only features used are the words in and around the mentions, their corresponding word vectors and the composed representations generated using the recursive autoencoder.

4.1 Stand-alone singleton detection performance

Evaluation showed that the system performance is slightly above the state-of-the-art (de Marneffe et al., 2015) on singleton detection in isolation and with the Stanford coreference resolution system, but that it performs clearly worse with the Berkeley coreference resolution system. After model optimization on the CoNLL-2012 development set, the system shows overall classification accuracies ranging from 76.34% on the 2011 development set to 80.08% on the 2012 test set. Performance on singleton mentions is consistently higher than on coreferent mentions, with F1-scores of 82-83% for singleton classification and F1-scores ranging from 64% to 76% for coreferent classification.

To achieve more precise classification, the classification threshold can be varied. This shows that singleton mentions can be identified with 90% precision at 55% recall. For coreferent mentions the recall at the same level of precision is around 50%, which shows that the system is capable of highly reliable classification, at a reasonable level of recall. For higher levels of precision, recall drops very fast.

The discrepancy between the performance on singleton mentions and coreferent mentions is an artefact caused by the binary nature of the classification task. Since the system assigns a label to every mention, it also classifies mentions for which it is not very certain ($P \approx 0.5$). Consequentially, it will always classify either too many mentions as singleton, or too many mentions as coreferentital.

The system could be optimized as to maximize performance on both singletons and coreferents, but this would require not labelling some mentions at all. Here, the system was trained to optimize overall accuracy, and this was achieved by classifying an overly large proportion of mentions as singleton. Because this discrepancy is not very meaningful, accuracy is taken here to be the best indicator of overall system performance.

The model was optimized with regard to four variables: hidden layer size, the number of context tokens in the input, the number of context mentions in the input and the set of vectors used for word representations.

For hidden layer size, there was no clear effect on the results. Overall, hidden layer size seemed to have only a small effect, and no consistent relations can be seen between the number of hidden nodes and the classification scores.

The most surprising result in optimization regards the effect of context tokens as features. Beforehand, we expected a reasonably large number of context words to yield the optimal effect, assuming that, up to a certain extent, more information leads to improved performance. On the contrary, it turned out that there is a consistent inverse relation between classification accuracy and the number of context words in the input. The model using only 1 word before and after the mention as input yielded the best performance, and with each additional word, accuracy dropped.

This contrasts with previous work: de Marneffe et al. (2015) use 2 words around the mention, and semantic information from a larger window. And in non-referential *it* detection, the context windows are generally larger, too, e.g. Bergsma and Yarowsky (2011) use up to 5 words before and 20 words after the *it* under consideration. Looking at the mention detection literature in general, we see that this pattern holds up: in non-referential *it* detection, larger context windows are used than in works that deal with complete NPs.

This provides a clue for how to explain the small context window used here. If a mention consists only of *it*, there is no mention-internal information to work with, and systems have to rely on contextual information for classification. With NPs, there is more mention-internal information, and contextual information is less important. Especially with larger NPs, the words around the mention can be quite far away from the semantic core of the NP.

It is likely that the same dynamic is at play with singleton detection. Even though this task covers both pronouns and NPs, the majority of mentions are NPs, which makes classifying them correctly most important for overall accuracy. The effect is exacerbated by the fact that OntoNotes only annotates the NP with the longest span if two or more NPs share a head word. This increases the average length of mentions in the corpus, which in turn makes the words around the mentions less relevant.

Another factor is the way in which context words are defined. De Marneffe et al. note that they only take context words from within the same sentence, and add padding tokens when context crosses sentence borders. Here, however, we chose to use padding only at document borders, and take context from other sentences too. The rationale behind this is that it adds information for mentions in short sentences, which would otherwise lack context. Additionally, it helps to implicitly capture positional information; if the first token in the left context of a mention is a ‘.’, this is a good indicator that the mention is sentence-initial. However, it might be that context words from outside the sentence containing the mention are less informational, which makes the incorporation of larger contexts detrimental to performance.

A third reason is that the context was taken symmetrically around the mention, i.e., the same number of words before and after the mention were used. The idea behind this is that, in singleton detection, classification depends both on antecendenthood and anaphoricity. Because of this, for some words, such as pronouns, the left context is more relevant, while for others, e.g. discourse-new NPs, the right context is more relevant. Therefore, the ratio of informative to non-informative words is lower for symmetrical contexts, which might hurt performance. Alternatively, it might be the case that left context or right context is more important overall, and this is not exploited by the current model. As such, a possible future improvement is to test different values for left and right contexts and evaluate the effects.

As far as the number of context NPs is concerned, the effect is less remarkable. Overall, the effect of this variable is small, and the optimal number seems to be 2 mentions before and after the mention under consideration. This seems reasonable, since mentions are most likely to be coreferential with those closest to them, and these mentions are thus the most relevant. For pronouns like *it* one would expect a relatively large number of NPs in the left-context to be the most beneficial. For larger NPs, the best option is not so clear. This is further obfuscated by the fact that mentions with large spans can contain other mentions, which are then also used as context mentions.

The effect of different sets of semantic word vectors on singleton detection performance is the largest of all variables tested, causing a difference of almost 2 percentage points. Overall, testing shows that higher-dimensional vectors result in higher accuracy. This is consistent with what the creators of the GloVe vectors, Pennington et al. (2014) found, for a different task.

They, and Collobert et al. (2011), also found that vectors trained on a larger corpus show better performance on a range of tasks. Here, by contrast, we see that the vectors trained on a 840 billion word corpus perform better than those trained on a 6 billion word corpus, but worse than those trained on a 42 billion word corpus. So, it is possible that, up to a certain size, bigger corpora do not yield any additional performance gains.

4.2 Performance as part of coreference resolution

The second part of system evaluation concerns assessing its performance when combined with coreference resolution systems. Previous work on mention detection (cf. Section 1.3.3) identified two factors that are important for effective mention filtering. First of all, precision in filtering is more important than recall, since filtering out non-singletons hurts resolution more than not filtering out singletons. Second, the way the filtering is integrated with the resolution system can make a crucial difference. Here, both were tested simultaneously, by using both regular and high-precision predictions in various manners in the coreference resolution systems.

For both the Berkeley and Stanford systems, we see that high-precision predictions boost performance the most. For the Stanford system, this is indicated by the good performance of models that filter singletons only for $P < 0.15$, and for Berkeley this is shown by the improvement gained by adding binary features for $P < 0.15$ and $P > 0.8$. As far as integration is concerned, we see that the filter for the Stanford system works best when it considers pairs of mentions, instead of single mentions. The same goes for the Berkeley system, which gains the most from a model that adds the raw probabilities and binary features for pairs.

De Marneffe et al. (2015) do not explicitly compare different ways of utilizing the singleton probabilities. However, their integration methods differ little from the best ones here; they too use probabilities and high-precision binary features for pairs for the Berkeley system, and a high-precision filter on pairs for the Stanford system. This confirms the findings here, that these are the optimal ways of utilizing singleton probabilities in these two types of coreference resolution systems.

The only difference regards the Stanford system. De Marneffe et al. found that not applying the filter to named entities improves performance, but here the opposite was found. A possible explanation for this is that the semantic word vector set used here also contains vectors for proper names, which allows them to be classified just like other words. These vectors contain semantic information for these proper names that might not be available in other approaches.

As for the coreference resolution performance with the optimized models, there is a clear discrepancy between the Berkeley and Stanford systems. For the Stanford system, we see a 0.7 point increase in CoNLL F1-score. Given that singleton detection accuracy is close to that of de Marneffe et al.'s system, we expect to see (at least) a similar effect on coreference resolution as they found. For the Stanford system, this holds, as de Marneffe et al. find a 0.5 point increase in CoNLL F1-score. For the Berkeley system, however, they report an increase of almost 2 percentage points. Here, however, we found only an insignificant increase of 0.3 percentage points in coreference resolution performance.

As shown in Section 3.3, this difference is mainly due to the lack of reduction in divided entity-type errors. Since both the overall singleton detection accuracy and the manner of incorporation in the resolution system are almost the same, the only explanation for the difference can be that there is a qualitative difference between our and de Marneffe et al.'s classifications. Somehow, their system covers information and provides predictions that the Berkeley system can utilize, while our system does not seem to add that in the same magnitude.

4.3 Project outcomes and conclusions

The results help answer one of the research questions of this project, namely: does the use of semantic word vectors yield a larger effect on coreference resolution performance, because they are an information source not utilized by coreference resolution and other mention filtering systems? Based on the results discussed earlier, the answer to this question has to be negative. Since, incidentally, the singleton detection accuracy of our system is almost the same as that of de Marneffe et al., we can make an almost direct comparison to their effectiveness for coreference resolution. Although the filter's effect on the Stanford system is slightly larger, the effect on the Berkeley system is a lot smaller. Therefore, we must conclude that the singleton detection system developed here is actually less effective as an improvement to a coreference resolution system, and this cannot be explained by difference in singleton classification accuracy.

However, there is success, too. Another goal of this project was to investigate whether semantic word vectors can be used for 'higher-level' NLP tasks, i.e. tasks that are more semantic in nature, rather than morphosyntactic. These vectors (combined with neural networks) have been shown to be highly effective for syntactic parsing and named-entity recognition, for example. In addition, they have been used in state-of-the-art systems for sentiment analysis, which is concerned with one very specific dimension of semantics, and paraphrase detection, which is mainly dependent on composition and sentence meaning.

Singleton detection, in this sense, should be seen as similar to paraphrase detection. It relies on the identification and composition of meaning for smaller units, sometimes single words, sometimes large NPs. But instead of doing a direct comparison, it is a more general classification task, since it aims to exploit this semantic information to assign the correct label. Unlike paraphrase detection, where the problem can be framed as 'Do these 2 sentences have the same meaning?', there is no such direct approach to the problem here. To determine whether something is singleton, one has to rely on both mention-internal meaning, contextual information and information on other mentions.

In conclusion, the answer is that yes, semantic word vectors can be used to perform more complex NLP tasks as well. In this project, we have shown that, with off-the-shelf word vectors and a very basic neural network, neither of which are specifically geared towards the task at hand, state-of-the-art singleton detection performance can be achieved. Crucially, unlike other approaches, it requires almost no background knowledge of the task, and most importantly, almost no feature engineering.

This has the added benefit that the system can be easily applied to other languages, since the system has almost no language-specifics. Additionally, since semantic word vectors can also be trained on raw text, the semantic word vector and neural network combination can also be applied to languages for which there are not many language resources readily available.

Despite the lack of task- and language-specific engineering of the system, it shows it can classify mentions correctly in 80% of the cases. This shows that, clearly, the general linguistic and background knowledge captured in semantic word vectors can be used to do all kinds of NLP tasks, from parsing and chunking to paraphrase and singleton detection. This makes them a simple to use, broad-coverage unit of knowledge. This aligns well with how we intended to use these vectors, namely as a directly usable piece of general knowledge, which can be applied to a specific NLP-task, similar to how humans use general knowledge to aid their linguistic understanding. This knowledge is usually covered by a combination of linguistic features like part-of-speech information, named entity tags, morphological information, semantic roles, etc. It would be interesting to see if semantic word vectors can be used directly in coreference resolution, and whether they can push performance higher.

Looking back at the application of this framework to singleton detection, it should be noted that there is a lot more to explore, if one wanted to try and improve singleton detection performance. Here, we strived to explore the neural network and semantic word vector approach for the first time, without adapting the system to the specifics of the task too much. Rather, it was attempted to show how a general approach can already yield good results.

4.4 Directions for future research

In order to make a better system, there are several options to consider. Singleton detection is a broad task that, unlike non-referential *it* detection, for example, covers all types of mentions and both antecedents and anaphors. Because of this, the system learns to classify all these things reasonably well, even though it might benefit from assigning more weight to certain features for the classification of certain mention types, for example.

An analysis of singleton detection performance, split out between pronouns, definite and indefinite NPs, mentions that start a cluster, and anaphoric mentions should shed light on whether there any categories for which the system does not perform well. A distinction between pronouns and NPs is easy to make. Perhaps training two separate systems, one for pronouns and one for NPs, could boost performance. For coreference resolution, the utility of this has already been shown by Hoste and Daelemans (2005), who separate pronouns, named entities, and other NPs. This seems to confirm the possible effectiveness of training separate models for singleton detection too, as the two tasks are so closely related.

Another possible improvement lies in the way mentions are represented. For all multi-word mentions, representations were generated using the RAE. Although this is a good compression method, it still loses a lot of information, especially for longer mentions. Information that is lost cannot be exploited by the network. The main reasons for applying the compression is to have fixed size input representations for mentions, regardless of their length, and to have more compact mention representations, which contain mostly relevant information, getting rid of non-informative words.

However, a classification system may benefit from having, instead of, or in addition to this information, word-level representations for mentions. This could consist of the word vector for the semantic head of the mention, and additionally a number of words around the head word. This way, a fixed-size input can be generated without compression. By focussing on the head word, the most important information is kept, and only peripheral information is lost.

As discussed in Section 2.1.1, a problematic part of the implementation of a singleton detection system has to do with the way 'mentions' are defined. Because the OntoNotes corpus does not annotate singletons in its coreferent annotations, the identification of what is, and what is not a mention is left to anyone who works on mention detection or coreference resolution. This makes sense for system evaluation, but it is not beneficial for system development and training.

Singleton detection in particular would benefit from having a pre-defined set of mentions. In this project, the mentions as defined by the Stanford coreference resolution system were used, but this does not mean that this is the correct set of mentions.

Not knowing whether a certain NP or pronoun is not a mention at all, or just a mention that is not coreferential and therefore not annotated is harmful for mention detection training, as it pollutes the training data. In addition, the absence of singleton annotation makes that every coreference resolution system uses a different set of mentions. As such, a mention filtering system would have to be adapted for each coreference resolution system to use the same set of mentions in order to attain maximal performance.

Looking at singleton detection as a task within the field of mention detection, it is clear that it is the most useful variant of the mention filtering tasks. It has the broadest scope and has shown the most promise when it comes to benefiting coreference resolution systems.

Regarding previous work in mention detection and filtering, two important observations can be made as to what would benefit future research. First of all, consistency in the use of definitions, resources and metrics is necessary. Looking at earlier research on non-referential *it* detection, for example, it is remarkable that each system is evaluated on a completely different corpus, with slightly different definitions and annotations of what constitutes non-referential *it*. This greatly hampers the ability to compare systems' performance to each other, which in turn prevents conclusions about what does and does not work for non-referential *it* detection. This is not just limited to non-referential *it* detection, but the same applies to other mention detection and filtering tasks.

The second note regards the manner of evaluation. Mention filtering has almost no use as a stand-alone application, but rather as a tool to be used in combination with other NLP systems. Therefore, evaluating the performance of a mention filtering system in isolation is not enough. Performance should be judged in-line, e.g. in combination with a coreference resolution system. Only then can the real value and quality of a system or task be judged. Fortunately, this has been done in several works (cf. Section 1.3.3), although not always with high-quality resolution systems. This has provided valuable insights, both for mention detection and coreference resolution.

All in all, this project aimed to bring together mention detection, and semantic word vectors and neural networks. As such, it forms both a small contribution to the field of mention detection and filtering, and a small exploration of the possibilities of the word vector and neural network paradigm. It is shown that, using off-the-shelf word vectors and a basic neural network, a high-performance singleton detection system can be built. Hopefully, this work inspires further research into mention filtering, which can be very useful for coreference resolution. Also, it should be taken as an impetus to explore the possibilities of neural networks and word vectors for natural language processing and computational linguistics, even though the approach is high on computation and low on linguistic intuitions.

References

Bagga, A. & Baldwin, B. (1998). Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference* (pp. 563–566).

Bean, D. L. & Riloff, E. (1999). Corpus-based identification of non-anaphoric noun phrases. In *Proceedings of ACL 1999* (pp. 373–380).

Bergsma, S., Lin, D. & Goebel, R. (2008). Distributional identification of non-referential pronouns. In *Proceedings of ACL-HLT 2008* (pp. 10–18).

Bergsma, S. & Yarowsky, D. (2011). NADA: a robust system for non-referential pronoun detection. In I. Hendrickx, S. L. Devi, A. Branco & R. Mitkov (Eds.), *Anaphora processing and applications* (pp. 12–23). Berlin: Springer.

Björkelund, A. & Farkas, R. (2012). Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the joint conference on EMNLP and CoNLL* (pp. 49–55).

Boyd, A., Gegg-Harrison, W. & Byron, D. K. (2005). Identifying non-referential *it*: a machine learning approach incorporating linguistically motivated patterns. In *Proceedings of the ACL workshop on feature engineering for machine learning in NLP* (pp. 40–47).

Brants, T. & Franz, A. (2006). Web 1T 5-gram Version 1 LDC2006T13.

Burnard, L. (2005). The BNC sampler, XML version. Retrieved from http://www.natcorp.ox.ac.uk/

Byron, D. K. & Gegg-Harrison, W. (2004). Eliminating non-referring noun phrases from coreference resolution. In *Proceedings of DAARC 2004* (pp. 21–26).

Cai, J., Mújdricza-Maydt, É. & Strube, M. (2011). Unrestricted coreference resolution via global hypergraph partitioning. In *Proceedings of the fifteenth conference on CoNLL* (pp. 56–60).

Chang, K.-W., Samdani, R., Rozovskaya, A., Rizzolo, N., Sammons, M. & Roth, D. (2011). Inference protocols for coreference resolution. In *Proceedings of the fifteenth conference on CoNLL* (pp. 40–44).

Chang, K.-W., Samdani, R., Rozovskaya, A., Sammons, M. & Roth, D. (2012). Illinois-Coremetrics UI system in the CoNLL-2012 shared task. In *Proceedings of the joint conference on EMNLP and CoNLL* (pp. 113–117).

Chen, C. & Ng, V. (2012). Combining the best of two worlds: a hybrid approach to multilingual coreference resolution. In *Proceedings of the joint conference on EMNLP and CoNLL* (pp. 56–63).

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of ICML 1995* (pp. 115–123).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

Daelemans, W., Zavrel, J., van der Sloot, K. & van den Bosch, A. (1998). *TiMBL: Tilburg Memory Based Learner, version 1.0, reference manual.* Technical Report ILK-9803. ILK, Tilburg University.

de Marneffe, M.-C., Recasens, M. & Potts, C. (2015). Modeling the lifespan of discourse entities with application to coreference resolution. *Journal of Artificial Intelligence Research*, *52*, 445–475.

Denis, P. & Baldridge, J. (2007). Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of NAACL-HLT 2007* (pp. 236–243).

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S. & Weischedel, R. (2004). The automatic content extraction (ACE) program - tasks, data and evaluation. In *Proceedings of LREC 2004* (pp. 837–840).

Durrett, G., Hall, D. & Klein, D. (2013). Decentralized entity-level modeling for coreference resolution. In *Proceedings of ACL 2013* (pp. 114–124).

Durrett, G. & Klein, D. (2013). Easy victories and uphill battles in coreference resolution. In *Proceedings of EMNLP 2013* (pp. 1971–1982).

Evans, R. (2001). Applying machine learning toward an automatic classification of *It. Literary and Linguistic Computing*, *16*(1), 45–57.

Fernandes, E. R., dos Santos, C. N. & Milidiú, R. L. (2012). Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the joint conference on EMNLP and CoNLL* (pp. 41–48).

Hirschman, L. & Chinchor, N. (1997). Muc-7 coreference task definition. In *Proceedings of MUC-7.*

Hoste, V. & Daelemans, W. (2005). Learning Dutch coreference resolution. In *Proceedings of CLIN 2004* (pp. 133–148).

Huddleston, R. D. & Pullum, G. K. (2002). *The Cambridge grammar of the English language.* Cambridge University Press.

Karttunen, L. (1976). Discourse referents. In J. D. McCawley (Ed.), *Notes from the Linguistic Underground* (Vol. 7, pp. 363–385). Syntax and Semantics. New York: Academic Press.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004* (pp. 388–395).

Kummerfeld, J. K. & Klein, D. (2013). Error-driven analysis of challenges in coreference resolution. In *Proceedings of EMNLP 2013* (pp. 265–277).

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M. & Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, *39*(4), 885–916.

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M. & Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the fifteenth conference on CoNLL* (pp. 28–34).

Litrán, J. C. C., Satou, K. & Torisawa, K. (2004). Improving the identification of non-anaphoric *it* using support vector machines. In *Proceedings of JNLPBA 2004* (pp. 58–61).

Luo, X. (2005). On coreference resolution performance metrics. In *Proceedings of HLT-EMNLP 2005* (pp. 25–32).

Mikolov, T., Karafiát, M., Burget, L., Černocký, J. & Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of Interspeech 2010* (pp. 1045–1048).

Mikolov, T., Yih, W.-t. & Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT 2013* (pp. 746–751).

Ng, V. & Cardie, C. (2002). Identifying anaphoric and non-anaphoric noun pphrase to improve coreference resolution. In *Proceedings of COLING 2002* (pp. 1–7).

Paice, C. D. & Husk, G. D. (1987). Towards the automatic recognition of anaphoric features in english text: the impersonal pronoun "it". *Computer Speech and Language*, *2*, 109–132.

Pennington, J., Socher, R. & Manning, C. D. (2014). GloVe: global vectors for word representation. In *Proceedings of EMNLP 2014* (pp. 1532–1543).

Poesio, M., Alexandrov-Kabadjov, M., Vieira, R., Goulart, R. & Uryupina, O. (2005). Does discourse-new detection help definite description resolution? In *Proceedings of IWCS 6* (pp. 236–246).

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O. & Zhang, Y. (2012). CoNLL-2012 shared task: modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the joint conference on EMNLP and CoNLL* (pp. 1–40).

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R. & Xue, N. (2011). CoNLL-2011 shared task: modeling unrestricted coreference in OntoNotes. In *Proceedings of the fifteenth conference on CoNLL* (pp. 1–27).

Quinlan, J. R. (1993). *C4.5: programs for machine learning.* San Mateo: Morgan Kaufmann.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 846–850.

Recasens, M., de Marneffe, M.-C. & Potts, C. (2013). The life and death of discourse entities: identifying singleton mentions. In *Proceedings of NAACL-HLT 2013* (pp. 627–633).

Recasens, M. & Hovy, E. (2010). Coreference resolution across corpora: language, coding schemes, and preprocessing information. In *Proceedings of ACL 2010* (pp. 1423–1432).

Recasens, M. & Hovy, E. (2011). BLANC: implementing the Rand index for coreference evaluation. *Natural Language Engineering*, *17*(4), 485–510.

Socher, R., Bauer, J., Manning, C. D. & Ng, A. Y. (2013). Parsing with compositional vector grammars. In *Proceedings of ACL 2013* (pp. 455–465).

Socher, R., Huang, E. H., Pennington, J., Ng, A. Y. & Manning, C. D. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS 2011* (pp. 801–809).

Socher, R., Lin, C. C.-Y., Ng, A. Y. & Manning, C. D. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML 2011* (pp. 129–136).

Socher, R., Manning, C. D. & Ng, A. Y. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS 2010 workshop on deep learning and unsupervised feature learning* (pp. 1–9).

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y. & Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP 2011* (pp. 151–161).

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y. & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP 2013* (pp. 1631–1642).

Song, Y., Wang, H. & Jiang, J. (2011). Link type based pre-cluster pair model for coreference resolution. In *Proceedings of the fifteenth conference on CoNLL* (pp. 131–135).

Uryupina, O. (2003). High-precision identification of discourse new and unique noun phrases. In *Proceedings of the ACL 2003 student research workshop* (pp. 80–86).

Uryupina, O. (2009). Detecting anaphoricity and antecedenthood for coreference resolution. *Procesamiento del Lenguaje Natural*, *42*, 113–120.

van Deemter, K. & Kibble, R. (2000). On coreferring: coreference in MUC and related annotation schemes. *Computational Linguistics*, *26*(4), 629–637.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D. & Hirschman, L. (1995). A model-theoretic coreference scoring scheme. In *Proceedings of MUC-6* (pp. 45–52).

Weischedel, R. & Brunstein, A. (2005). BBN Pronoun Coreference and Entity Type Corpus. Retrieved from https://catalog.ldc.upenn.edu/LDC2005T33

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., ... Houston, A. (2011). OntoNotes Release 4.0. Retrieved from https://catalog.ldc.upenn.edu/LDC2011T03

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., ... Houston, A. (2013). OntoNotes Release 5.0. Retrieved from https://catalog.ldc.upenn.edu/LDC2013T19

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of COLING 2000* (pp. 947–953).