# IMPROVING COREFERENCE RESOLUTION BY USING NOUN-PHRASE CLUSTERS

Bachelorproject

Rik van Noord, s2067560, R.I.K.van.Noord@student.rug.nl

Jennifer Spenader

**Summary:** Coreference resolution is the task of determining whether two noun phrases refer to the same entity in the real world. In this study we use automatically generated noun clusters as a semantic information source in our supervised machine learning approach to solve the task of (Dutch) coreference resolution, as was done in Hendrickx et al. (2008), using the clustering method of van der Cruys (2005). We investigate the effect of cluster size (average amount of words per cluster) and test the effect of the clusters on a dataset of only common nouns. We try to find an optimal cluster size and hypothesize that the optimal average size of a cluster will be higher than the average size (10) of the clusters used in Hendrickx et al.(2008). Adding cluster features yielded a 27,9% higher F-score than our baseline, and by testing different cluster sizes we found an optimal cluster size of 10 words per cluster, with slightly lower F-scores the more we move away from 10 words per cluster. The results suggest that an average size of 10 words per cluster is optimal for the task of coreference resolution.

## 1. Introduction

Coreference resolution is the task of resolving different descriptions in natural language to the same underlying entity in the real world. Consider the following example:

> **(1) Theo Maassen** is een **cabaretier** uit **Eindhoven**. De **entertainer** houdt zijn eerste show in **Groningen**.
> **English: Theo Maassen** is a **comedian** from Eindhoven. **The entertainer** is hosting his first show in **Groningen.**

In this example, *Theo Maassen*, *cabaretier ('comedian')* and *entertainer ('entertainer')* all refer to the same entity in the real world, namely *Theo Maassen* himself. For language processing systems, such as question answering and summarization, understanding and solving coreference resolution is a necessary challenge. Over the last years, viewing coreference resolution of noun phrases (NP's) as a classification task that can be solved by supervised machine learning has become increasingly popular. We view coreference resolution as a binary classification task, which means that for every pair of NP's, the task of the classifier is to determine whether the NP pair is coreferential or not (positive or negative). The binary approach requires an annotated corpus of training documents, with coreferential links between NP's. Next, instances are created between every NP (possible anaphor) and all of its preceding NP's (possible antecedent), within a scope of 3 sentences. These instances are used as training and test examples in the form of a feature vector, and are given to a classifier. The task of the classifier is determining if a pair of NP's is coreferential or not. The feature vector for supervised machine learning approaches usually consists of a combination of lexical, positional, syntactic and semantic information sources. Especially semantic information sources seem quite useful for the task, since we know that people base their decisions strongly on *real world knowledge*. However, most current coreference systems don't make much use of semantic information, while we know humans use this information when solving coreference. There are many different ways to encode semantic information. In this study, we focus on adding features from a specific semantic information source, namely automatically generated semantic clusters of Dutch NP's. This has been done before by Hendrickx et al. (2008) by using noun-clusters made by Van Der Cruys (2005), but they only tested clusters with an average of 10 words per

clusters. We investigate the effect of average cluster size and hypothesize that that the optimal average size of a cluster will be higher than an average of 10 words per cluster, as was used in Hendrickx et al. (2008).

**1.1 Clusters**

The observation that a noun is a member of a particular cluster might not be very informative. The observation that both nouns belong to the same cluster might be strong evidence of a coreferential relation. Consider the following clusters used by Hendrickx et al. (2008), table 1:

**Table 1: An example of the noun clusters, made by Van der Cruys (2005).**

| | |
|---|---|
| **127** | Cabaretier, clown, entertainer, goochelaar, komiek, stand-up_comedian, strateeg<br>**English:** Comedian, clown, entertainer, magician, comic, stand-up_comedian, strategist |
| **598** | Amsterdam, Den_Bosch, Den_Haag, Eindhoven, Groningen, Haarlem, Leiden, Rotterdam, Tilburg, Utrecht, Zwolle |
| **615** | aanvaller, aanvoerder, captain, doelman, exinternational, international, keeper, libero, middenvelder, routinier, spits, sterspeler, verdediger<br>**English:** attacker, captain, captain, goalkeeper,exinternational, international, keeper, libero, midfielder, veteran, striker, star player, defender |
| **765** | Gaston_Taument, John_de_Wolf, Orlando_Trustfull, Peter_Bosz, Regi_Blinker, Rob_Witschge, Ronald_Koeman, Standard_Luik, Ulrich_van_Gobbel |
| **867** | Clarence_Seedorf, Danny_Blind, Edgar_Davids, Edwin_van_der_Sar, Finidi_George, Frank_Rijkaard, Jari_Litmanen, Marc_Overmars, Patrick_Kluivert, Peter_van_Vossen |
| **917** | Asprilla, Baggio, Beckenbauer, Charlton, Cruijff, Gullit, Keizer, Maradona, Pele, Van_Basten, Walter, Wilkes |

Consider example (1) again. In this example, *cabaretier* and *entertainer* corefer and they're both a member of cluster 127. It seems that knowing that two NP´s are a member of the same cluster would be useful for coreference resolution. It would be even more useful if we consider that many of the words in cluster 127 can corefer with each other. For instance, *cabaretier* and *stand-up comedian ('stand-up comedian')*, *clown ('clown')* and *entertainer* or *komiek ('comic')* and *entertainer*. But, the same sentence also shows why having this information can hurt. Eindhoven and Groningen do not corefer, but are a member of the same cluster. In fact, the cluster only exists of Dutch cities, which will never corefer with each other. A more informative source might be the combination of clusters. Consider the following example (2):

**(2)** **Clarence Seedorf** won de **Champions League** vier keer. De **international** was echter nooit erg geliefd in **Nederland**.
**English:** **Clarence Seedorf** won the **Champions League** four times. **The international** was, however, never really popular in **The Netherlands.**

Cluster 867 is a list of successful Dutch soccer players from Ajax and cluster 615 is a cluster of words that often corefer with (successful) soccer players. In this example, it will help classifying *Clarence Seedorf* and *international ('international')* as coreferential. The combination of those two clusters is strong evidence of a coreferential relation. This is also explained in Hendrickx et al. (2008).

Hendrickx et al. used clusters made by Van der Cruys(2005), but they did not experiment with the average number of words per cluster. Van der Cruys did not create the clusters specifically for the goal of coreference resolution. As far as we know, optimal cluster size has never been systematically tested with the coreference resolution task.

Now consider the clusters 765 and 917. Cluster 765 is a list of successful players from Feyenoord and cluster 917 exists of iconic soccer players over the last 50 years. Because they both exist of soccer players, they also corefer often with cluster 615. In this case, the classifying algorithm needs to learn three relational clues, namely that the combination of clusters 615-756, 615-867 and 615-917 are a clue that the noun-pair might be coreferential. If those clusters are added

together, the classifier doesn't have to learn three strong relational clues, but it only has to learn one. This will make the learning process faster and it will improve the performance of the classifier. If we use a higher average of words per clusters than used in Hendrickx et al.(2008), it seems likely that all soccer players will be clustered together. This is not only the case for soccer players, but also for cities, musicians, companies and other examples where certain nouns can best be put in one cluster. A disadvantage might be that cluster 615 also will get bigger, resulting in a cluster of words that can corefer with soccer players, but also of words that only corefer with tennis or baseball players. This disadvantage might be overcome by the fact that we use an average of words per cluster and not an exact number for every cluster. It's perfectly possible to obtain multiple clusters of 5 or 60 words if we use an average of 20 words per clusters. So, it stands to reason that similar words like soccer players will end up in bigger clusters than the more dissimilar words that are used to refer to all kind of sportsmen. In this paper, we try to find an optimal cluster size and hypothesize that using a higher average of words per cluster might be beneficial for the automatic parsing of coreference resolution using machine learning.

### 1.2 Related work

We will shortly describe some history of (Dutch) coreference resolution. Machine learning solutions for coreference resolution have received quite some attention over the last 18 years. Mcarthy and Lehnert (1995), Soon et al. (2001) and Ng and Cardie (2002) desribe in their papers a supervised machine learning solution for coreference resolution. The method of Soon et al. (2001), a decision tree learning approach, was the first machine learning based solution that could offer performance comparable to non-learning approaches, while only using 12 features. A lot of work has been put into using semantic features; especially WordNet (Miller (1995)) remains a useful information source (Poesio et al. (2004) and Harabagiu et al. (2001)). WordNet is a lexical database of English words and labels the semantic relations among words. It contains a lot of semantic information, including information about synonyms, antonyms, hypernyms and hyponyms. These types of semantic information are really useful for coreference resolution. Ponzetto and Strube (2006) used hierarchical semantic knowledge mined from WordNet and Wikipedia and used it to create a measure of semantic similarity. They obtained a 14,3% and 13% improvement in accuracy for common nouns for both features. However, Markert and Nissim (2005) described that the knowledge encoded in WordNet is often insufficient and they described a Web-based method that outperforms WordNet for *other*-anaphora. Ng (2007a) explored the use of automatically extracted semantic classes and found significant (2 – 6%) improvement in F-score. Ng (2010) wrote a review article, summarizing the progress made since 1995 on the subject of supervised coreference resolution and describing a shift from classifiers using only local information to more global, expressive classifiers that can exploit certain cluster-like features of nouns that already coreferred.

For Dutch, Hoste and Daelemans (2004) created a machine learning system using a K-NN classifier and using the Flemish Knack-2002 corpus, the same one we will be using, using 17 features and dividing the data in pronouns, proper nouns and common nouns. Hendrickx et al. (2008) added semantic information sources using the automatically generated semantic clusters made by Van der Cruys (2005) along with EuroWordNet features (Vossen, 1998) and showed, with an F-score of 46.45 for their baseline, a small increase in performance for adding cluster features (F-score 47.11), no increase for only WordNet features (F-score 46.33) and a bigger increase for using both WordNet and cluster features (F-score 47.45).

## 2 Method

### 2.1 Features

There are several information sources that contribute to solving coreference resolution, for example syntactic, morphological, lexical, positional and semantic information sources. Because we want to know the impact of adding the noun clusters, we have to create a baseline. The baseline only exists of information we can directly extract from the actual sentences in the annotated corpora. We use 7 features also used

in Soon et al. (2001) and 5 features used by Hoste and Daelemans (2005). All used baseline features are shown and explained in Table 2.

**Table 2. A description of the baseline features (information sources) used in this study.**

| Feature | Description | Possible values |
|---|---|---|
| DIST-SENT | Number of sentences between anaphor and antecedent | 0,1,2,3..20 |
| DIST-LT-THREE | DIST-SENT smaller than 3? | yes, no |
| NUM-AGREE | Do the antecedent and anaphor agree in number? | yes, no |
| GEN-AGREE | Do the antecedent and anaphor agree in gender? | yes, no |
| SAME-NE | Do anaphor and antecedent have the same NE-type (person, organisation or location?) | yes ,no |
| BOTH–SUBJ-OBJ | Do the antecedent and anaphor agree in syntactic relation? | Subj, obj1, obj2, .., no |
| STR-MATCH | Is one of the candidates a substring of the other? | yes,no |
| LEFT-WD1 | The word before the anaphor, if possible, otherwise NA | Word, NA |
| LEFT-POS1 | Part Of Speech tag of the word in LEFT-WD1, if possible | POS, NA |
| J-DEF | Is the anaphor demonstrative? | yes, no |
| J-DEM | Is the anaphor definitive? | yes, no |
| APPOS | Is the coreferential noun phrase in apposition to the NP preceding it? | yes,no |

**2.2 Clustering**

To create noun-clusters, it's best to use as much data as possible. The more information we have about a particular noun, the more certain it is that it ends up in the right cluster. The noun clusters are created by using the Lassy Large corpus. Lassy Large is a corpus composed from multiple corpora (Eindhoven, Wikipedia, Europarl, Sonar500) and contains 700 million words. Lassy is automatically annotated, based on a manually annotated corpus of 1 million words.

Noun clusters for Dutch data were made before by Van der Cruys (2010), using the Twente Nieuws Corpus. In his dissertation, he carefully lays out a method for evaluating semantic similarity and creating noun clusters. We followed this method in our study and we'll carefully describe it in the next paragraphs.

Van der Cruys (2010) hypothesizes in his dissertation that words are similar if they appear in similar syntactic contexts and that the syntactic context of a particular word is suitable to inform us about its semantics, as was described by Firth (1957). In his research he uses dependency graphs; a theory-neutral instantiation of syntax, since no underlying grammatical framework is assumed. Dependency relations describe the relation between two words in a sentence, one being the head and one being the dependent. A number of relations that are useful for (distributional) semantic similarity models are adjectives, direct object, indirect objects and subject-verb relations. For a detailed description of the relations, I refer to Van der Cruys (2010).

Dependency relations are useful, because instead of just looking at all the words in the environment of a particular noun, we look at the syntactic relation every noun has with the words in its environment. For instance, a word that occurs a lot as the direct object of *drinken* is probably a drink, but if another word occurs a lot as the subject of *drinken,* it's probably someone or something that drinks a lot. Every noun has a lot of those relations with other words and lot of the specific relations will occur more than once. All this specific information is put in a vector. All these vectors combined result in a noun-by-features matrix. The vector is then normalized by vector length, to facilitate

computations. A simplified example of the non-normalized resulting noun-by-features matrix is shown in table 3.

**Table 3. A simplified example of a non-normalized noun-by-features matrix.**

| | Groen(adj) | Mooi(adj) | Eten (obj) |
|---|---|---|---|
| Peer | 112 | 3 | 98 |
| Gras | 275 | 17 | 31 |
| Auto | 30 | 165 | 1 |
| Grap | 0 | 8 | 0 |

A problem that arises here is the fact that the very frequent, broad features, such as *mooi ('beautiful')* or *lekker ('delicious')*, get more weight attached to them then the less frequent, but more meaningful features such as *katoenen ('cotton')* of *smakelijk ('tasty')*. For example, the adjective *katoenen* probably only occurs with some kind of clothes and if we see *smakelijk* as an adjective, the word is probably some food. While *mooi* and *lekker* can also occur with those types of words, they also occur with multiple other types of words, for example; *mooi nummer ('beautiful song'), mooi meisje ('beautiful girl'), mooi huis ('beautiful house')*, *lekker weer ('delicious wheather'), etc*. So the co-occurrence of nouns with broad, frequent words as *mooi* , *lekker* or *leuk ('nice')* doesn't tell us much about the semantic information of the noun. A co-occurrence like *katoenen* or *smakelijk* is much more informative, even though the other broad adjectives are more frequent.

A way to solve this problem is the use of a weighting function. Van Der Cruys uses the Pointwise Mutual Information (PMI) function, as was first proposed by Church and Hanks (1990). Mutual information is a quantity that measures the mutual dependence between two random variables X and Y. Pointwise Mutual Information is the mutual information between particular events and is calculated as is shown in equation 1, with *i* and *j* as events.

$$I(i,j) = \log \frac{p(i,j)}{p(i)p(j)}$$

**Equation 1. The calculation of Pointwise Mutual Information, from Van der Cruys (2010).**

PMI measures how often two certain events co-occur, compared to the expected value if it were two independent events. The final ratio shows how much more the noun and feature co-occur than we would expect by chance. This function is thus able to attach more weight to the infrequent, but more informative features in our noun-by-features matrix.

The noun-by-features matrix is complete, so we can now focus on the clustering. The actual dividing of the features vectors into a number of clusters is done by clustering software Cluto (Karypis, 2003). The final solution of the dataset divided in *K* is obtained by performing a sequence of *K*-1 repeated bisections. This means that the data is divided into two groups, and then the most dissimilar of the groups is divided further. This is done in such a way that the resulting 2-way solution is optimized. This process continues until *K* clusters are found. Note that this process ensures that the clustering is locally optimized within each bisection, but not globally. That's why the overall solution is globally optimized after the initial bisections. There is not much known about the optimal average cluster size, so we decided to look at extremes too. We clustered our noun-by-features matrix into 9 different cluster-files, with an average of 2, 3, 5, 10, 15, 20, 30, 40 and 50 words per cluster.

### 2.3 Cluster features

To encode the semantic information of the noun clusters, we use the features *sameCluster* and *(cluster1,cluster2).* The possible values for *sameCluster* are 0 or 1 and *cluster1* and *cluster2* are the cluster numbers the antecedent/anaphor is in, or *NA* if the NP isn't part of a cluster. This is shown in Table 4.

**Table 4. The cluster features used in this study.**

| Feature | Description | Possible values |
|---|---|---|
| sameCluster | Do the anaphor and antecedent belong to the same cluster? | 0,1 |
| c1, c2 | Cluster numbers of the antecedent and anaphor, or NA | Numbers or NA |

A problem with the *sameCluster* feature is that it replictates the work of the string-match feature most of the time. If two words are exactly the same, by definition they must be in the same cluster. This matters especially for clusters with a small average size. If *sameCluster* gets the value 1 and the possible anaphor and antecedent are an exact string-match, both these features convey the same semantic information. Because of occurrences like this, the classifier might misjudge the influence of the *sameCluster* feature when the possible antecedent and anaphor are not exact string matches, but do belong to the same cluster.

Hendrickx et al. (2008) do not make clear whether they included *sameCluster = 1* for exact string matches. For the *isHyponym* and *isSynonym* features they also use, they describe in a footnote that exact string-matches are considered synonyms and hyponyms. The K-NN classifier TiMBL (further described in section 2.3), which we are going to use for classifying, has built-in mechanism to eliminate features with the same informational value, but it might still influence classifying slightly. It will definitely skew the calculations of the influence of each feature. To avoind making sketchy assumptions, we will create every dataset twice, once with *sameCluster = 1* and once with *sameCluster = 0* for exact string matches.

**2.4 Experiment**

To create a trainingset and testset, we use the KNACK-2002 corpus, coreference annotated by Hoste and Daelemans (2005), to create the dataset with positive and negative instances. KNACK-2002 is a Flemish weekly news magazine, which writes about current national and international affairs, such as politics, sports, business and similar other news topics. The whole corpus contains 267 documents in which 12546 nouns phrases are annotated with coreferential information. Cluster features only seem useful for solving coreference resolution for common nouns, because pronouns and to a lesser extent proper nouns, do not carry much (if any) semantic information. That is why our dataset only exists of instances with common nous. To form the test instances, all NP's starting from the second NP are possible anaphors, and all preceding NP's are possible antecedents. If we combine every two possible NP, this will result in an enormous dataset, consisting of virtually only negative instances. To reduce the amount of negative instances, we use a scope of 20 sentences to form NP-pairs and label them as positive or negative, as was seen in Hoste and Daelemans (2005). This led to a dataset of 129182 instances of common nouns, in which 3491 instances are labeled positive, with the added features as shown in Table 2.

Using the 9 different clusters-files, we created 9 different datasets. For every cluster-file, the *sameCluster* and *cluster1,cluster2* features will be different. The specific features are added to our baseline. Because the baseline is the same for every cluster, testing the performance of the classifier will test the impact of the different clusters on the performance of the classifier. Every dataset is trained and tested separately.

For the actual K-NN classifying, we used the memory-based-learning system TiMBL, developed by Daelemans et al. (2004). TiMBL contains a collection of memory-based machine learning methods, including K-NN. A key feature of memory based machine learning is that all examples are stored and that no attempt is made to simplify the model by eliminating noise, low frequency events or exceptions. In language learning tasks, like coreference resolution, it's very hard to discriminate between noise and valid exceptions (which are important for good accuracy). Daelemans, van den Bosch and Zavrel (1999) found that editing exceptions in language memory based learning tasks is harmful. They found that abstraction or editing of the model is never beneficial for generalization accuracy. We chose TiMBL specifically because we want to compare our results to Hendrickx et al. (2008) and Hoste and Daelemans (2005), who both used TiMBL's K-NN classifier.

K-NN is a non-parametric method of classifying new, unseen data, based on the most similar training examples in the training data. K-NN is a simple machine learning method, the new instance is classified by a majority vote of the class of its *K* nearest neighbors. In this study, we use *K*=1, which is the same as Hoste and Daelemans (2005) and Hendrickx et al. (2008), and is a fairly safe choice according to Cover and Hart (1967). This means that the test instance is assigned to the same class (positive or negative) as his nearest neighbour. The rest of the parameters are left at their default values[1]. We used 10-fold cross validation to test the performance of the classifier. 10-fold cross validation divides the dataset in 10 equally large parts. It first trains on 9 parts and then tests on the part left out. For each of our datasets, all consisting of baseline features and additional different cluster features, we conducted a 10-fold cross validation test. Because we use 9 different cluster sizes, this resulted in 9 evaluations of the classifier, and thus 9 different evaluations of the clusters. This was done twice for the different measures of the *sameCluster* feature, as was explained in section 2.1.

**2.5 Evaluation**

The evaluation is done by *precision* and *recall* of the positive examples, ultimately leading to an F-score. This is calculated only based on the classifications of positive instances, because of the overwhelming number of negative instances. In our dataset, 3491 of the 129182 instances were labeled positive. This means that by simply guessing negative every time, we would achieve an accuracy of 97.3%. Thus calculating accuracy based on all data would unrealistically drive up the score. We avoid this problem by only calculating the F-score based on the classifications of positive instances. This is common in coreference resolution and was also done by Hendrickx et al (2008) *Precision* is calculated by dividing the number of correctly classified positive instances by the number of instances classified positive. *Recall* is calculated by dividing the number of correctly classified positive instances by the total amount of positive instances in the dataset. The F-score is a combination of both *precision* and *recall* and is obtained by the equation shown in equation 2. F-scores are calculated for each of the 9 datasets created by adding different cluster features.

$$\text{F score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

**Equation 2. The calculation of the F-score**

## 3 Results

**Table 5. The difference between with and without string-match for the sameCluster feature, based on our 9 feature sets.**

| | **Instances with sameCluster = 1** | |
|---|---|---|
| **Average clustersize** | ***sameCluster=0* for exact string-match** | ***sameCluster=1* for exact string- match** |
| **50** | 509 | 2597 |
| **40** | 489 | 2577 |
| **30** | 503 | 2591 |
| **20** | 336 | 2424 |
| **15** | 304 | 2392 |
| **10** | 337 | 2425 |
| **5** | 230 | 2318 |
| **3** | 158 | 2246 |
| **2** | 62 | 2150 |

Our results are split in two parts; F-scores for all cluster features with *sameCluster = 1* for exact string-matches and F-scores for all cluster features with *sameCluster = 0* for exact string-matches. To show the difference between *sameCluster = 1* or *sameCluster = 0* for exact string-matches, we calculated the number of times *sameCluster* was assigned the value 1 for each of our datasets, as is shown in Table 5. Only 11.9% percent of the time (on average), *sameCluster* is assigned the value 1 due to different words in the same cluster, e.g. where exact matches are assigned *sameCluster = 0*. This matters especially for lower average sizes. For example, this percentage drops to 3.0% for an average size of 2 words per cluster. The other 97% percent of the time (and on average 88.1%), *sameCluster* is assigned 1 due to exact string-

[1] Hoste (2005) showed that coreference resolution could benefit from parameter optimization, this is however not essential to our goal of comparing which average cluster size is best.

matches. This shows why we decided to conduct every experiment twice, with *sameCluster = 1* or *sameCluster = 0* for exact string-matches.

Table 6 shows the precision, recall and F-score for every average size of words per cluster used for creating the 9 different cluster feature sets, with the *sameCluster = 1* for exact string-matches. Table 7 shows the precision, recall and F-score for the 9 datasets, with *sameCluster = 0* for exact string-matches.

**Table 6. Precision, Recall and F-score for the different averages of cluster sizes, *sameCluster =1* for exact string-matches.**

| Average clustersize | Precision | Recall | F-score |
|---|---|---|---|
| **Baseline** | 0.3866 | 0.2212 | 0.2814 |
| **50** | 0.3969 | 0.3286 | 0.3595 |
| **40** | 0.4061 | 0.3300 | 0.3641 |
| **30** | 0.3989 | 0.3266 | 0.3591 |
| **20** | 0.4024 | 0.3306 | 0.3630 |
| **15** | 0.4077 | 0.3323 | 0.3662 |
| **10** | 0.4164 | 0.3311 | 0.3689 |
| **5** | 0.4045 | 0.3294 | 0.3631 |
| **3** | 0.4147 | 0.3309 | 0.3681 |
| **2** | 0.4094 | 0.3283 | 0.3644 |

**Table 7. Precision, Recall and F-score for the different averages of cluster sizes, *sameCluster =0* for exact string matches.**

| Average clustersize | Precision | Recall | F-score |
|---|---|---|---|
| **Baseline** | 0.3866 | 0.2212 | 0.2814 |
| **50** | 0.3957 | 0.3286 | 0.3590 |
| **40** | 0.4063 | 0.3297 | 0.3640 |
| **30** | 0.4013 | 0.3280 | 0.3610 |
| **20** | 0.4009 | 0.3286 | 0.3611 |
| **15** | 0.4066 | 0.3311 | 0.3650 |
| **10** | 0.4166 | 0.3277 | 0.3668 |
| **5** | 0.4083 | 0.3309 | 0.3655 |
| **3** | 0.4124 | 0.3245 | 0.3633 |
| **2** | 0.4093 | 0.3231 | 0.3611 |

Adding cluster features drastically improves performance from a baseline F-score of 0.2841 to an average F-score of 0.3635. This huge improvement is mostly due to an improvement in *recall* from 0.2212 for the baseline to an average of 0.3286 for adding cluster features. *Precision* only improved from 0.3866 for the baseline to an average of 0.4064 for adding cluster features.

For both *sameCluster = 1* and *sameCluster = 0* for exact string-matches, an average cluster size of 10 led to the best score. For *sameCluster = 0* for exact string-matches, the second and third best F-score were achieved by an average cluster size of 5 and 15. For *sameCluster = 1* for exact string-matches, the second and third best score were achieved by an average size of 3 and 15 words per cluster. The results suggest that moving away from an average cluster size of 10 will hurt the performance of the classifier. Although, in both cases, an average cluster size of 40 obtained a better F-score than an average cluster size of 20 or 30 words per cluster. Also, the worst score was obtained by an average cluster size of 50. The average F-score for *sameCluster =1* for exact string-matches (0.3640) and *sameCluster =0* for exact string-matches (0.3630) only shows a very small difference. So, adding cluster features benefits the automatic parsing of coreference resolution a great deal, but using a higher average size than 10 words per cluster does not seem to improve the classifying.

**Table 8. Gain ratio of the two cluster features, for every average cluster size, *sameCluster =1* for exact string-matches.**

| Average cluster size | cluster1, cluster2 | sameCluster |
|---|---|---|
| **50** | 0.0054 | 0.1081 |
| **40** | 0.0056 | 0.1086 |
| **30** | 0.0057 | 0.1082 |
| **20** | 0.0060 | 0.1175 |
| **15** | 0.0061 | 0.1195 |
| **10** | 0.0063 | 0.1187 |
| **5** | 0.0065 | 0.1258 |
| **3** | 0.0067 | 0.1282 |
| **2** | 0.0068 | 0.1343 |

**Table 9. Gain ratio of the two cluster features, for every average cluster size, *sameCluster = 0* for exact string-matches.**

| Average cluster size | cluster1, cluster2 | sameCluster |
|---|---|---|
| **50** | 0.0054 | 0.0026 |
| **40** | 0.0056 | 0.0022 |
| **30** | 0.0057 | 0.0025 |
| **20** | 0.0060 | 0.0040 |
| **15** | 0.0061 | 0.0047 |
| **10** | 0.0063 | 0.0065 |
| **5** | 0.0065 | 0.0116 |
| **3** | 0.0067 | 0.0068 |
| **2** | 0.0068 | 0.0120 |

Table 8 and 9 show the gain ratio of the two cluster features per average cluster size. To calculate the gain ratio, TiMBL first calculates the information gain per feature independently and calculates how much it contributes to our knowledge of the correct class label. Information gain tends to overestimate the impact of features with a large number of values. To solve this, TiMBL calculates the *gain ratio*, a relative normalized *information gain* measure. Because TiMBL calculates this for each feature independently, it doesn't take into account that a lot of the *sameCluster* information will be redundant if we assign 1 to *sameCluster* for exact string-matches. Table 9 shows the real influence of the *sameCluster* feature, where we assign 0 to *sameCluster* for exact string-matches. The ranking of the gain ratios differ slightly per cluster size, but *sameCluster* and *cluster1,cluster2* are usually the fourth and fifth best feature. *STR-MATCH*, *SAME-NE* and *DIST-SENT* usually outperform the cluster features, but the cluster features do have a higher gain ratio than the also informative features *GEN-AGREE* and *NUM-AGREE.* For both cluster features, we see an inversed correlation between gain ratio and cluster size, instead of a correlation between F-score and gain ratio. This might be due to fact that gain ratio has an unwanted bias towards features with more values, as was shown in White and Liu (1994). This favours the smaller clusters, because smaller clusters result in more values for the *cluster1,cluster2* feature.

## 4 Discussion

In this paper we investigated the effect of adding information about membership in automatically generated semantic noun clusters as features to improve coreference resolution. We tried to find an optimal cluster size and hypothesized that using a higher average cluster size than 10 (as was used in Hendrickx et al. (2008)) will improve automatic coreference resolution. Adding cluster features showed a big improvement, our baseline results improved from a baseline F-score of 0.2841 to an average F-score of 0.3635 (27,9%).

We also found an optimal size of 10 words per cluster. For both the string-match and no string-match values for *sameCluster* an average size of 10 words per cluster was the optimal size for automatic coreference resolution. The rest of the results suggest that the more we move away from an average cluster size of 10, the more it will hurt the performance of the classifier. So, Hendrickx et al. (2008) did have the right empirical solution, without testing. The results do now show that using an higher average cluster size than 10 is beneficial for automatic coreference resolution. The exception seems to be an average cluster size of 40 words per cluster, which in both cases scores better than an average of 20 and 30 words per cluster. The string-match and no string-match values for *sameCluster* show only a small difference, so it seems that TiMBL succeeded in eliminating redundant information.

It's hard to compare our F-score to other research, most research is done for English instead of Dutch, evaluation metrics differ per research, and moreover, most research doesn't just test common nouns. Hoste and Daelemans (2005) got an overall F-score of 51.4 for proper nouns, common nouns and pronoun and a precision of 47.6 for common nouns. They do use a couple of very useful semantic and string-match features, that weren't employed in this study, because of the information available at the time of the parsing. Hendrickx et al. (2008) received an F-score of 45.6 for all three noun types, after adding only the cluster features. But, they also used a richer set of syntactic features (e.g. clause information, dependency paths) than was used in this study.

The gain ratio drastically changes from an average of 0.1181 for *sameCluster* replicating the string-match (about as high as the real string-match) feature to 0.0058 for *sameCluster* not replicating the string-match feature. So, for calculating the gain ratio, TiMBL doesn't take the redundancy of information into account. We see the real influence of the sameCluster feature if it doesn't replicate the string-match feature, as was shown in Table 9. But, as we explained in the results section, gain ratio has an unwanted bias toward features with more values. The lower the amount of average words per cluster, the higher the amount of different features will be for *cluster1, cluster 2*. This is probably why the gain ratio of *cluster1,cluster2* increases as the average cluster size gets lower. The gain ratio shows an inversed correlation to cluster size, instead of a correlation with F-score, so all we can conclude is that both the cluster features have a positive influence on coreference resolution, but the exact impact of a single feature is hard to determine.

So, although we found an optimal cluster size, the results did not show that using a higher average number of words per clusters is beneficial for coreference resolution. When clustering more words together on average, semantically dissimilar words sometimes end up together, because they have to be assigned to a cluster. The disadvantage of clustering together semantically dissimilar words exceeded the advantage of adding together different clusters with members of the same type, such as cities, soccer players, musicians, etc. Adding dissimilar words to a cluster also decreases the informative value of the *sameCluster* feature, because words in the same cluster do sometimes not have a semantic relation at all, especially for an average size of 30 words or higher per cluster. Clusters could be optimized manually, but then we are in danger of manually creating a WordNet-like database, instead of empirically finding the semantic relations between nouns. We might miss or manually edit information provided by the clustering algorithm that seems counter intuitive, but is useful for coreference resolution. This is explained in example 3.

**4.1 Analysis**

To show the impact of the cluster features, three examples extracted from our test data are shown in which we explain how adding the cluster features led to a different result than our baseline.

**(3)** Tussen de 11de en de 15de eeuw verkochten kooplui uit alle hoeken van Europa in **Brugge** een waaier van goederen, gaande van basisproducten tot exclusieve luxewaren. **De stad** is altijd een internationale handelsplaats geweest.

**English:** Between the 11th and 15th century, merchants from all over Europe sold in **Brugge** a range of goods, ranging from basic to exclusive luxury goods. **The city** has always been an international marketplace.

**Table 10. Two clusters based on an average cluster size of 10 words per cluster.**

| | |
|---|---|
| **221** | Knokke, Damme, Middelkerke, Koekelare, Diksmuide, Blankenberge, Koksijde, Beernem, Nieuwpoort, Jabbeke, Gistel, Brugge, Ichtegem, Kortemark |
| **943** | regio, richting, district, station, spoor_lijn, hartje, stad, gewest, provincie, bisdom, arrondissement, gemeente, woon_plaats<br>**English:** region, direction, district, station, railway, center, city, region, province, diocese, district, town, hometown |

Consider example 3. This is a positive instance, because *stad ('city')* en *Brugge* both refer to the same entity in the real world. Our baseline algorithm classified this example as negative, while the classifier made the correct decision if the two cluster features were added based on an average of 10 words per cluster. The algorithm correctly learned that words in cluster 221, Flemish villages or small cities, often corefer with words in cluster 943, words often used to refer to villages or cities. This example shows why synonym or hyponym information may fall short and why manually editing the clusters is dangerous. For example, *richting ('direction')* and *provincie ('province')* aren't words an editor might manually cluster together. Cluster 943 certainly

doesn't contain only synonym or hyponym information, yet all words are often used to corefer with villages or cities, so for this task we want the classifier to have this information. However, sometimes the clustering information will hurt the performance. Consider example 4.

**(4)** De kenmerkende **verpakking** van het Heinekenbier, het groene **blikje** met de rode ster en de schuine, lachende letter , werd door hemzelf ontworpen.

**English:** The distinctive **packaging** of the Heineken Beer, the green **can** with the red star and the oblique, laughing letter, was designed by himself.

**Table 11. Two clusters based on an average cluster size of 15 words per cluster.**

| | |
|---|---|
| **92** | liter, slof, blik, glas, pint, fles, druppel, tube, voorraad, zak, pak<br>**English:** liter, carton, can, glass, pint, bottle, drop, tube, stock, bag, pack |
| **605** | indeling, ophaling, geheugen, verpakking, opslag, afzet, afvoer, remmer, inzameling, teelt, transport<br>**English:** classification, collection, storage, packaging, storage, disposal, discharge, inhibitor, collection, cultivation, transportation |

This is a positive example, as *verpakking ('packaging')* and *blikje ('can')* both refer to the same entity in the real world, in this sentence. If we add the cluster features for an average cluster size of 15, the classifier classifies this example as negative. Apparently, there was a negative example in our training data in where the nouns in the noun-pair both belonged to exactly the same cluster. In that way, it ended up being the nearest neighbour and the classifier made the mistake to label example 4 negative, the same as the nearest neighbour. The algorithm learned a wrong negative clue. This is the case for a lot of examples. It's not the case that by adding cluster features, a few examples will now be classified correct and that's all. A lot of instances will now indeed be classified correct, but a lot of instances will also now be classified wrong. This is partly an effect of sparse data. If we had more data, then the nearest neighbour will be a better representative of the NP-pair we're testing, so fewer instances will go from classified correct to classified wrong. This will differ for the different average cluster sizes. Consider example 5.

**(5)** Dit culmineerde in de manier waarop hij in het Canvas-programma TerZake voor schut kwam te staan door het bestaan te ontkennen van een **brief** die hij zelf had getekend - nota bene het enige **document** dat hij in dit dossier te tekenen kreeg. Vlak voor de uitzending had Van Hemelrijck hem bevestigd dat zo'n **brief** niet bestond.

**English:** This culminated in the way he made a fool of himself in the Canvas-program TerZake, by denying the existence of a **letter** he had signed himself – take notice this was the only **document** in this file he received to sign. Just before broadcact Van Hemelrijck had confirmed to him that such a **letter** didn't exist.

**Table 12. Two clusters based on an average cluster size of 10 words per cluster.**

| | |
|---|---|
| **151** | Dreigbrief, mail, telefoontje, email, sms, fax, brief, bericht<br>**English:** threating letter, mail, call, email, sms, fax, letter, message |
| **878** | Informatie, document, notitie, raadgeving, memo, gegeven, suggestie, info, rapport, inlichting, advies<br>**English:** information, document, note, counsel, memo, data, suggestion, information, report, intelligence, advice |

**Table 13. Two clusters based on an average cluster size of 2 words per cluster.**

| | |
|---|---|
| **674** | brief, bericht<br>**English:** letter, message |
| **733** | document, rapport<br>**English:** document, report |

Consider the example with *document ('document')* and the second occurrence of *brief ('letter').* In this example, the baseline classifies this positive instance incorrectly as negative. After adding cluster features with an average cluster size of 2, it still classifies this instance incorrectly. When adding cluster features with an average of 15 words per cluster, the classifier makes the correct

decision. There was apparently a positive example which had the exact same clusters for a cluster size of 10, but a cluster size of 2 missed this example because it was too small. For example, if the positive instance had a noun-pair of *email ('email')* and *informative ('information')*, this helps classifying other noun-pairs from clusters *151* and *878* as well for a cluster size of 10. Because of the small average clusters, an average cluster size of 2 puts *email* and *informatie* in different clusters than *brief* and *document*. Now the classifier can't use this information and it makes the wrong decision. The same logic applies for the big average clusters, with an average of 30 or higher, but now with the opposite effect. If the clusters are too big, too many unrelated words get clustered together and the classifier extracts wrong information for words that are in the same cluster, but are semantically unrelated. Examples like this explain the difference between the F-scores for the different average sizes of the clusters.

Now consider example 5 again, but now with the first occurrence of *brief* and *document* as an instance. This gets classified correct by our baseline, but the other *brief* and *document* instance gets classified wrong. This is due to the fact that we view coreference resolution as a binary classification task. Instead of attempting to find coreference relations between noun-pairs, it might be better to link NP's to existing coreferential clusters (Yang et al. 2003). Instead of the semantic information of just a single noun, we can use the more rich semantic information of the noun-cluster the noun belongs to. These clusters are made in the training phase and can be seen as a chain of nouns who refer to the same entity in the real world. This broader representation of a single noun will help the classifying. It might have solved the example with the first occurrence of *document* and *brief*, because then we already knew that *document* and *brief* referred to the same entity before.

**4.2 Future research**

Coreference resolution is still far from solved. Ng (2010) states in his review paper of 15 years of supervised machine learning coreference resolution that *mention-pair models* (those who view coreference resolution as a binary classification task) are weak, because they make decisions based on only local information. Mention-pair models fail to answer the question which of the possible antecedents is the most probable. It doesn't express how likely a candidate is relative to other candidate antecedents. A better option is using entity-mention models, as explained in Yang et al. (2004) and also as a more probabilistic implementation by McCallum and Wellner (2003). Ng (2010) concludes that expressive models that can exploit cluster-like, non-local features offer better performance. So it seems that coreference resolution can benefit from shifting the attention from viewing coreference solution as a binary decision task to a more global task with feature that are able to exploit non-local features and can keep track of existing coreferential chains.

There are a lot of different semantic information sources before which have been used for coreference resolution. All these features differ from each other, but when added together, they'll show a lot of redundancy. We agree with Ng (2010), who states that researchers should use a better baseline system than Soon et al. (2001) and make their work publicly available, so that performance comparisons might be possible. In this way, we'll be able to see which semantic features are new and which semantic features are redundant combinations of features used in other research. We want to arrive at an optimal level of semantic information, instead of combining all possible semantic features to obtain the best results.

The previous paragraph described the future of coreference resolution as a whole. The feature for automatically generated semantic clusters might be limited. Yang et al. (2003) showed that the viewing coreference resolution as a binary classification task might not be optimal. A combination of automatically generated noun-clusters and keeping track of coreference chains existing of more than two words would be interesting to see. Maybe even a combination of the two clusters might be possible, creating clusters of words that already coreferred to each other and words that are likely to do so. The clustering itself seems hard to improve, now that we see that average cluster length doesn't show much room for improvement. It might be that the way we made

the clusters has an effect on the performance. We focussed completely on a rich variety of dependency information between words, while we didn't do anything with broader information about the environment in which the word occurs. For example, looking at the topic (e.g., politics, sports, economics) of sentences, paragraphs or documents in which the word appears might be helpful for creating a better feature vector for a particular noun.

An interesting option also might be an extra cluster feature, indicating whether nouns in a certain cluster are able to corefer at all. As we saw in example 1, this is important, because certain clusters have members that can never corefer (*Groningen, Eindhoven, Rotterdam*), but other clusters do have members that are able to corefer (*cabaretier, entertainer, komiek)*. It's maybe possible to extract this information using the same type of data as we used now as train and test datasets. A disadvantage of this approach is that we need more data than we already need, because we can't use the same data for creating the cluster features as we use for training and testing. This is due to the obvious danger of overfitting our system on a particular dataset. However, having this information might be beneficial for improving the cluster features and is an interesting subject for further research.

A combination of the semantic clusters with other semantic features, such as synonym/hyponym information (i.e. Hoste and Daelemans (2005)), semantic similarity (Ponzetto and Strube (2006)) or semantic class agreement as was used in Soon et al. (2001), might also be an interesting way to go. Automatically generated semantic clusters might show connections between words that other semantic information sources might miss, as was explained after example 3 in section 4.1.

## 5 Conclusion

In this paper, we used automatically generated semantic clusters to improve coreference resolution. Specifically, we tried to find an optimal cluster size and hypothesized that the optimal average size of a cluster will be higher than the average size (10) of the clusters used in Hendrickx et al. (2008). We found a 27,9% improvement in F-score for adding the cluster features, and found by testing different cluster sizes that the optimal average size of the clusters is around 10 words per cluster, with the F-score decreasing the more we move away from an average of 10. Automatically generated semantic clusters might be useful for future work, but probably only to improve more promising semantic information sources.

## References


Church, Kenneth Ward and Patrick Hanks. 1990. *Word association norms, mutual information & lexicography*. Computational Linguistics, 16(1):22–29.

Daelemans, Walter, Antal van den Bosch, and Jakub Zavrel, *Forgetting exceptions is harmful in language learning*. In: Machine Learning, 34:1/3, 1999, pp. 11-43.

Daelemans,, W., Zavrel, J., Van der Sloot, K., Van den Bosch, A.: *TiMBL: Tilburg Memory Based Learner, version 5.1, reference manual.* Technical Report ILK-0402, ILK, Tilburg University (2004)

T. M. Cover and P. E. Hart. *Nearest neighbor pattern classication.* Information Theory, IEEE Transactions on, 13(1):21{27, 1967.

Van de Cruys, T.*: Semantic clustering in Dutch*. In: Proceedings of the Sixteenth Computational Linguistics in the Netherlands (CLIN). (2005) 17{32

Tim Van de Cruys. 2010. *Mining for Meaning. The Extraction of Lexico-semantic Knowledge from Text.* PhD thesis. University of Groningen, The Netherlands.

J. Firth. 1957. A synopsis of linguistic theory 1930-1955. *Studies in Linguistic Analysis, Philological Society, Oxford*, reprinted in Palmer, F. (ed. 1968) Selected Papers of J. R. Firth, Longman, Harlow.

Harabagiu, S., Bunescu, R., Maiorano, S.: *Text and knowledge mining for coreference resolution*. In: Proceedings of the 2nd Meeting of the North American Chapter

of the Association of Computational Linguistics (NAACL-2001). (2001) 55{62

Hendrickx Iris, Véronique Hoste, Walter Daelemans. '*Semantic and syntactic features for coreference resolution for Dutch*.' In: Proceedings of the CICLing-2008 Conference, Haifa, Israel, Berlin: Springer, 2008, p. 351- 361.

V. Hoste. *Optimization issues in machine learning of coreference resolution*. PhD thesis, University of Antwerp, 2005.

Hoste, V., & Daelemans, W.(2005). *Learning Dutch coreference resolution* . Proceedings of the Fifteenth Computational Linguistics in the Netherlands Meeting (CLIN 2004).

Karypis, George. 2003. CLUTO - *a clustering toolkit*. Technical Report #02-017, nov.

Markert, K., Nissim, M.: *Comparing knowledge sources for nominal anaphora resolution.* Computational Linguistics 31(3) (2005) 367{401

George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

Ng, V.*: Semantic class induction and coreference resolution*. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Association for Computational Linguistics (2007a) 536{543

Vincent Ng.: *Supervised Noun Phrase Coreference Research: The First Fifteen Years*. Main Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10), 2010.

Poesio, M., Mehta, R., Maroudas, A., Hitzeman, J.: *Learning to resolve bridging references*. In: Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04). (2004) 143{150

S. P. Ponzetto and M. Strube. *Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. pages* 192{199. Association for Computational Linguistics, 2006.

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

White, A.P. and W.Z. Liu. 1994. *Bias in information-based measures in decision tree induction*. Machine Learning, 15(3):321–329.

X. Yang, J. Su, G. Zhou, and C. L. Tan. 2004. *An NPcluster based approach to coreference resolution*. In Proc. of COLING, pages 226–232.