Computational Linguistics in the Netherlands Journal 5 (2015) 15-24 Submitted 06/2015; Published 11/2015

Automatic word sense disambiguation for Dutch using dependency information

Hessel Haagsma HESSEL.HAAGSMA@RUG.NL

Rijksuniversiteit Groningen, Oude Kijk in 't Jatstraat 26, 9712EK Groningen, The Netherlands

Abstract

An automatic word sense disambiguation system utilizing dependency information is implemented using existing language resources for Dutch (Lassy, Alpino, Cornetto) and tested on a subset of DutchSemCor. The disambiguation method used largely follows the method first proposed by Lin (1997). It defines words by their local context, represented as dependency triples. The notion that words occurring in the same local contexts are semantically close to the ambiguous word is used to create a list of similar words. The correct sense is then found by selecting the sense that is semantically closest to the words in this list.

Performance on a set of nouns, verbs and adjectives is tested, and overall performance is comparable or slightly higher than that reported by Lin: almost 9% over baseline for fine-grained sense distinctions and over 3% over baseline for coarse-grained sense distinctions. In absolute terms, disambiguation accuracy was highest for nouns, slightly lower for verbs and lowest for adjectives. The effect of using different local contexts and semantic databases was tested, which indicated that a reliable sense-annotated corpus is still required and that quality and types of dependency relations in the local context database matters more than quantity. Overall, performance is as expected, showing that dependency contexts are a useful feature for word sense disambiguation for Dutch.

1. Introduction

Lexical ambiguity is a universal of language and a problem for every NLP application that deals with meaning. For example, an automatic Dutch-to-English translator translating the word *verbouwen* is posited with the choice of translating this as either *cultivate* or *renovate*. In order to solve this problem, a distinction has to be made between *words* and *senses*. Ambiguous words, such as *verbouwen*, are said to have multiple distinct senses or meanings.

Translating senses instead of words would prevent translation troubles like the one illustrated above. However, before translation of senses can be done, one would have to know which sense a certain word in a certain text has, and preferably determine this automatically. This is what constitutes the task of *word sense disambiguation*, which we will be concerned with in this study.

The research field of word sense disambiguation is well-developed and many different methods and approaches have been proposed. Here, we focus on one method, which utilizes dependency knowledge. The two most important works on this topic are by Lin (1997) and by Chen et al. (2009).

Lin provides a concise formulation of the motivation behind the use of dependency knowledge: "Two *different* words are likely to have *similar* meanings if they occur in *identical* local contexts." In practice, this means that, to find the sense of a word in a certain context, the sense that is closest to other words occurring in the same context is likely to be the right sense.

The key concepts here are *similar meanings*, which requires a measure of similarity between senses and *local contexts*, which requires a definition and representation of the context of a word. For the first, a semantic database can be utilized, while the local contexts can be represented by syntactic dependency structures. In order to then find other words that occur in the same local context, a large database of contexts and the words that occur in them is needed. Although acquiring this

database requires a large corpus, the main advantage of this approach is that it does not require the corpus to have sense annotations.

Lin (1997) was the first to develop a dependency based method, and his algorithm follows these steps: first, a database of local contexts is created by extracting so-called *relation triples* from a corpus. These triples consist of two words, the relation between them and an indication of which word is the head. For example *boy: subject-chase-head*, indicates a relation in a sentence where 'boy' is the subject of 'chase'. Then, from this database, the words occurring in the same contexts as the ambiguous word are retrieved. Finally, the sense of the original ambiguous word most similar to the meanings of the set of similar words from the database is selected as the answer.

The algorithm presented by Chen et al. (2009) largely follows the same principles, with some exceptions. Instead of using a pre-existing corpus, they collected text from the internet, using the to-be-disambiguated-word as an anchor term for retrieval. Another difference lies in the similarity measure: instead of using the hierarchical structure of the semantic database, they used the definitions of each possible sense of the ambiguous word and parsed these. This was then matched to the dependency structure of the sentence in which the to-be-disambiguated word occurred. A third difference is that Chen et al. used unlabelled relations, i.e. they did not use the relation type as an information source.

The algorithm presented here closely follows the original algorithm by Lin, in an attempt to assess the value of dependency-based word sense disambiguation for Dutch. As such, the main difference will lie in the language and the language-specific resources used. In addition, different context databases, a broader test set and different methods of scoring will be evaluated and discussed.

2. Resources

For the disambiguation algorithm to work, certain databases have to be created, and for that, linguistic resources are needed. The first database, the local context database, requires a large corpus, preferably one with syntactic dependency annotations. Fortunately, such a corpus exists for Dutch, in the form of the Lassy corpus (van Noord et al. 2013). Lassy contains almost 700 million words, which have been annotated automatically using the Alpino parser (van Noord 2006). Since Lin achieved good results using only a 25-million-word corpus, using the Lassy corpus is suitable for achieving equally good or better results.

The second resource required is a semantic database, which provides senses for words and relations between senses. Both the senses and the relations between them are needed to calculate a similarity measure for senses. A recent and comprehensive semantic database for Dutch exists, by the name of Cornetto (Vossen et al. 2013a). Cornetto 2.0 contains 92 thousand lemmas (nouns, verbs and adjectives), 118 thousand senses and 70 thousand synonym sets (synsets), most of which are connected by hierarchical relations to at least one other synset.

In addition to the semantic database, a corpus with reliable sense annotations is required as a test set. Preferably, the annotations should use the senses from the semantic database, so no additional mapping has to be made from one type of sense annotation to another. For this purpose the DutchSemCor (DSC, Vossen et al., 2011) corpus is used, which contains 1 million sense annotations, of which 250 thousand have been made manually. One of its advantages is that the annotations in the DSC use the same senses as Cornetto. Another plus of the DSC is that it partly draws on the same material as Lassy, and that this material is aligned with the Lassy corpus itself. This allows for easy and fast retrieval of dependency structures for the sentences containing ambiguous words.

In order to be able to make use of these resources, they have to be preprocessed into different databases. The largest, and arguably most important, database is the context database, a collection of relation triples and the count of their occurrences in Lassy. This database was constructed using a simple script that processed the whole Lassy corpus. It retrieved the dependency structures of each sentence in the corpus, and used a depth-first search, searching through each tree structure, extracting relations. Only relations containing either a noun, a verb, an adjective or an adverb were

extracted, on the assumption that other relations, e.g. those between nouns and their determiners, are uninformative. The resulting database contains 72 million relations of 8.6 million different types (i.e. each relation triple occurred in the corpus almost 8.5 times on average).

In order to test the influence of different context databases on performance, a second knowledge base was created using an alternative method. This database was created using Alpino, the same parser that was used for the syntactic parsing of the corpus. The Alpino parser is more advanced than the script, as it finds triples that are not directly present in the dependency tree. This results in two additional types of triples: relations between two words joined by a conjunction, and relations between each verb of a verb phrase and the subject and object. In the other database, only the relation between the head verb (often an uninformative auxiliary verb) and the subject and object was included. For example, for the sentence "Zij hebben koekjes en taart gegeten", this results in the addition of the triples *koekjes-taart-conjunction*, and *zij-gegeten-subject*.

The Alpino-produced database differs on one more aspect: in addition to the noun, verb, adjective and adverb, relations containing prepositions were also included. However, as all these additional relations make the database extremely large, only triples that occurred in the corpus at least 4 times were included. This resulted in a final database containing 1.1 billion relations, of 26.8 million different types, an average of 41 occurrences for each triple.

A different type of database to be created is the semantic database, from which a measure of semantic similarity between different synsets can be calculated. To do this, a hierarchy of synsets is needed, in addition to a count of how commonly a synset is used. Unfortunately, the Cornetto database does not contain a hypernym-hyperonym relationship for every synset. For nouns and verbs, almost all sets are part of at least one such relationship, but for adjectives, only about 50% is part of the hierarchy. As a result, synonym sets that are not part of any relation cannot be taken into account in the disambiguation process.

Three different ways of calculating the frequency of occurrence for synsets were used. The first method does not count the number of synset occurrences in a corpus, but uses the total number of synonyms in the synset and its hyperonyms as an estimation of frequency. The second variant does use a corpus, namely the manually annotated part of the DSC that overlaps with Lassy. The frequency count for each synset was calculated by summing the occurrences in the corpus of each sense in the synset and its descendants. The third variant is identical to the second, with the exception that it uses both the manually and automatically annotated part of the Lassy-part of the DSC. This has the advantage of having a lot more annotations, but the disadvantage of being less accurate in its annotations.

In addition to the context and semantic databases, one other resource is needed, a test set. Although the DSC can be used as a test set as is, it contains far too many annotations for testing purposes. Therefore, a test set of 942 items was created by equally distributed sampling from the manual annotations of the DSC. Consequentially, the majority of test items originates from the largest sub-corpora (each sub-corpus represents one type of text, e.g. newspapers), just like most of the relations in the context database are derived from the largest sub-corpora.

3. Algorithm

When all necessary resources have been created, the disambiguation process can start. The algorithm to do this is largely the same as that by Lin (1997) and is summarized below.

First, the dependency structure of the sentence containing the ambiguous word is retrieved from Lassy. From this dependency tree, all relation triples are extracted in the same way the context database was created, using either Alpino or the script written by the author. Then, only those relations containing the to-be-disambiguated word are selected.

The second step is to find other words that occur in the same context. For each relation in which the ambiguous word occurs, all words occurring in the same relation in the context database are retrieved from the context database. For each of these words, the log-likelihood-ratio- or G^2-score

is calculated, as described by Dunning (1993). This score is a measure of how much the frequency of a word in a context deviates from its expected frequency. If the score is high, this indicates that the word is typical of that context, and not just a highly-frequent word overall. Optionally, a frequency-based cut-off is implemented for contexts with a large number of entries in the database and a penalty is applied to their G^2-score. The details and effect on performance of this are discussed in the next section. After scores have been calculated for all words in all contexts, the scores for each context are added up and the highest-ranking words are selected.

The third and final step is then to retrieve all synsets to which these highest-ranking words belong from the semantic database. Additionally, all synsets to which the to-be-disambiguated word might belong are retrieved too. Then, for each synset containing a high-ranking word, the semantic similarity to each synset containing the ambiguous word is calculated using the log-likelihood-ratio as defined by Lin (1997, p.67). Using the counts from the semantic database, this results in a score for the similarity between each pair of synsets. For each synset containing the ambiguous word, the similarity to each synset containing a similar word is summed. The synset that has the highest score is selected as the answer and compared to the annotation in the DSC.

4. Evaluation and Results

The performance of the word sense disambiguation algorithm was tested on the test set of 942 instances using different resources and parameters. Testing was done on a regular desktop computer and took between 60 and 120 minutes. The test set contained not only nouns, but also verbs and adjectives. Of the 942 words, 552 were nouns, 328 were verbs and 62 were adjectives.

Besides regular accuracy scores, the similarity between the correct sense and an incorrect answer was also calculated, yielding a good indication of how far off an incorrect answer is. It is also used to produce a secondary accuracy score. In his paper, Lin argues that the senses in his semantic database are too fine-grained, and presents a second score that is, in his opinion, closer to the score that would be achieved if regular dictionary sense distinctions were used. Here, the same threshold value of a similarity score of 0.27 will be used, i.e. all senses with a similarity score over 0.27 are considered to be the same in the coarse-grained sense distinction condition.

Unless explicitly mentioned otherwise, the tests reported here used the Alpino context database, the DSC (manual) frequency counts, no G^2-score weighting, the 20 most similar words and no limit on the number of entries per context in the database. The baseline score was calculated by taking the first sense from the list of possible senses for the to-be-disambiguated word as the answer.

Context Database	**Accuracy** ($sim. = 1$)	**Accuracy** ($sim. > 0.27$)
Baseline	28.24	**61.58**
Script (7M word corpus)	29.51	58.60
Script (70M word corpus)	32.27	60.83
Script (350M word corpus)	31.32	**61.46**
Script (700M word corpus)	31.32	59.66
Alpino (700M word corpus)	**33.76**	60.30

Table 1: Word sense disambiguation accuracy (%) using different context databases. The whole of the DSC was used to derive frequency counts.

Table 1 shows the effect of using different context databases on performance. In addition to the two databases discussed previously (labelled here as Alpino and Script (700M)), three other databases are tested. These three databases are based on sets of sub-corpora from Lassy. The sets are composed to contain approximately the number of words indicated between brackets. Immediately,

two things stand out: the database based on 7 million words performs clearly worse than the others and the database derived using Alpino performs better than the others. We can also see that, no matter the database, the algorithm performs above the baseline score for fine-grained sense distinctions. This is slightly better than the original algorithm by Lin, who reported a score slightly below baseline using the strictest accuracy criterion.

Oddly enough, considering the different 'Script'-databases, the 70-million words set performs better than the larger sets, which is incongruous with the general trend of 'more data is better'. Looking at the coarse-grained (similarity over 0.27) score, we see that the algorithm classifies almost 60% of words as either correct or 'almost correct'. This score is close to, but slightly lower than the baseline score.

Semantic Database	**Accuracy** ($sim. = 1$)	**Accuracy** ($sim. > 0.27$)
Baseline	28.24	61.58
Synonym counts	30.68	**63.38**
DutchSemCor (manual)	**36.41**	63.34
DutchSemCor (all)	33.76	60.30

Table 2: Word sense disambiguation accuracy (%) using different semantic databases. DutchSemCor (all) represents the case where the whole of DSC was used for frequency counts, DutchSemCor (manual) where only the manual annotations of the DSC were used. Synonym counts represents the case where synonym counts instead of occurrence counts in a corpus were used.

In addition to the effect of the context database, the effect of the frequency count method on performance was also tested. Table 2 shows the influence on performance of the three different frequency count methods. Perhaps surprisingly, the smaller, but more accurate manual part of the DSC proves to be a better source for frequency counts than the DSC as a whole. The difference is sizeable, almost 3 percentage points, indicating that the frequency count is as crucial as the context database and that accuracy is more important than size. As expected, the database using synonym counts performs the worst. Clearly, the number of synonyms in a synset is not a good indicator of the number of actual uses of those synonyms.

The coarse-grained accuracy for the synonym counts database is equally good as the best-performing database, but this is likely a spurious result. Because the similarity measure is dependent on the semantic database, the high coarse-grained accuracy is not caused by actual better disambiguation performance, but by an overall larger similarity between the synsets in that database.

Synset Score Weighting	**Accuracy** ($sim. = 1$)	**Accuracy** ($sim. > 0.27$)
Baseline	28.24	61.58
No weighting	**36.41**	**63.34**
G^2-score weighting	36.20	62.63

Table 3: Word sense disambiguation accuracy (%) with and without G^2-score weighting.

Table 3, 4 and 5 show the influence on performance of three different changes in the algorithm. The first, Table 3, shows the effect of weighting the synset scores by the G^2-score of the highest-ranking similar words. This means that the similarity between the synset of the highest-ranking word and the synsets to which the ambiguous word belongs has a stronger influence than the second-highest-ranking word, and so on. The intuition behind this is that the highest-ranking word is also

Number of similar words	**Accuracy** ($sim. = 1$)	**Accuracy** ($sim. > 0.27$)
Baseline	28.24	61.58
10	35.56	62.21
20	36.41	63.34
40	**37.15**	63.91
80	36.84	**64.23**

Table 4: Word sense disambiguation accuracy (%) using different numbers of highest-ranking similar words.

Entry limiting method	**Accuracy** ($sim. = 1$)	**Accuracy** ($sim. > 0.27$)
Baseline	28.24	61.58
None	**37.15**	63.91
Frequency threshold	36.84	**64.65**
Context size penalty	36.84	64.33

Table 5: Word sense disambiguation accuracy (%) using two methods of limiting the number of entries a certain context can have. In the 'Frequency threshold'-case, all contexts with more than 5000 entries are reduced to a smaller number by implementing a frequency threshold: only the 5000 most frequent entries are included in calculations. In the 'Context size penalty'-case, no entries are discarded, but larger contexts are weighted to have less influence than contexts with less entries. The number of highest-ranking similar words used was 40.

the most similar and should thus carry the most weight. However, it seems that this is not the case, as the effect on performance of the weighting is minimal, with both fine- and coarse-grained accuracy being slightly lower than the scores for the algorithm without weighting.

The second table, Table 4 illustrates the effect of selecting more or less similar words. The value used in earlier tests, 20, is the one used by Lin, but seems to be chosen somewhat arbitrarily. Given that the context database used here is several times larger than the 25-million word corpus used by Lin, a larger number of similar words might prove to be beneficial. The results show that, indeed, using a larger set of similar words benefits performance: a set of 40 words has the highest fine-grained accuracy and using a set of 80 words yields the highest coarse-grained accuracy. Overall, using a set of 40 similar words seems to be better than using a set of 20 words, but the performance gain is small and for practical purposes, this might not be worth the extra computational cost.

The final algorithm variations tested are two different methods for limiting the influence of contexts with a large number of entries in the database, in order to test the assumption that larger contexts are less informative. Additionally, limiting large contexts improves processing speed. Again, the influence on performance is minor. The simple measure that takes only the most-frequent entries from large contexts seems to perform the best, but still has a slightly lower fine-grained accuracy than the original algorithm. As such, the implementation of entry limits is not beneficial to performance, but should be considered for applications where speed is key.

5. Discussion

The current study presents a first attempt at word sense disambiguation for Dutch using dependency knowledge. An algorithm similar to that described by Lin (1997) was successfully implemented and showed good results. An accuracy of almost 9 percentage points over the first-sense baseline was achieved. Looking at the accuracy on a more coarse-grained disambiguation task, approximated using the semantic similarity measure, an accuracy of almost 65% was achieved, 3 percentage points above the baseline (5). Testing was done on nouns, as well as adjectives and verbs, where Lin only tested nouns. On nouns, the accuracy was 40.58%, on verbs, 36.42% and on adjectives, 23.40%. Performance on verbs was close to that on nouns, but adjectives proved to be more difficult. This can be directly explained by the fact that many adjectives are not part of the hierarchical structure in the semantic database, yielding less accurate similarity measures for adjectives. It is also possible that word sense disambiguation difficulty in general varies by part-of-speech, since Vossen et al. (2013b) also found better performance for nouns, when compared to verbs and adjectives.

Comparing the results to that of Lin in an absolute manner, we see that the performance achieved here is almost 20 percentage points lower, as is the baseline. The reason for this is that the test set used here was drawn from the manually annotated part of the DSC, which was set up in such a way that each sense of a synset occurs approximately equally often in the corpus. This is different from other corpora and language in general, where different senses of a word tend to vary largely in terms of their frequency. As such, one could say that the task of disambiguation using items from the DSC is harder than it would be when a different corpus were used, and therefore, performance should not be judged in absolute terms, but relative to the baseline.

The current research attempted to explore the effect of using different resources and parameter values on the performance of the algorithm. It was shown that the largest context database performed best, but also that the use of a smaller databases does not dramatically decrease performance. This suggests that, although the use of larger corpora and databases might improve disambiguation accuracy, it is likely not the area where the highest performance gain is to be expected. A similar conclusion can be drawn from the test of different frequency counts for the semantic database. Here, the frequency count based on a smaller, but more accurate sense-annotated corpus proved to benefit performance the most. The difference in performance here was quite large, suggesting that a larger, manually annotated corpus might improve performance, especially if a corpus and test set are used which show a natural distribution of sense frequencies, which the DSC does not. An alternative

explanation is that the frequency distributions of different senses from the manually annotated part of the corpus better matches that of the test set, than the counts based on the non-sense-balanced automatically annotated part of the corpus. The counts in the semantic database partially reflect this, which could help improve performance.

The other manipulations, which were small adaptations to the algorithm, showed only limited influence on performance. The G^2-score weighting and limits on large contexts did not show any improvement and can be deemed irrelevant, although the latter might be useful for speeding up the disambiguation process. The use of larger sets of similar words had a slight positive effect, suggesting that, using a large context database, there might be more than 20 relevant similar words in some cases. Using even larger context databases might make this effect larger. Of course, there are inherent limitations on the amount of similar words a specific sense can have; languages only have a certain number of words for a set of related concepts.

Looking at the results of the best-performing test run in more detail, the specific strengths and weaknesses of this disambiguation algorithm become clear. Out of 942 ambiguous words, 350 were assigned the exact correct sense. On average, each word had 4.6 senses, showing the difficulty of the task and the scope of the Cornetto database.

For most words, an informed attempt was made to provide a sense answer, but for 42 words, no local context was found in the sentence. For 6 more words, contexts could be found, but no entries for these contexts could be found in the context database. In such cases, the disambiguator cannot do anything, and simply guesses from the possible senses of the word. In some cases, there was indeed no context, for example with the word *zalig*, found in the sentence 'Zalig!'. In other cases, due to the wide coverage of the corpus, the sentence in which the ambiguous word occurred was not a well-formed sentence and the parser could not derive any relations containing the to-be-disambiguated word.This is an obvious area for improvement, e.g. by improving the parser coverage or by incorporating context outside the scope of the sentence. However, the overall effect would be minor, as only 4.5% of words were complete guesses.

More room for improvement can be found when looking at the performance on different parts-of-speech. Results showed that disambiguation performance for adjectives was the poorest. A straightforward solution for this would be to improve the hierarchical relation annotation in the semantic database, which is likely the cause of the low performance on adjectives.

Although not as low as the performance on adjectives, performance on verbs was also lower than that for nouns. A possible explanation for this is that verbs do not lend themselves as well to a hierarchical structure as nouns do. Intuitively, verbs conform less to a 'part-of' or 'kind-of' structure than nouns. To improve this, the most obvious solution would be to improve the way the semantic database is utilized. Instead of exploiting only hypernym-hyperonym relations for the similarity measure, other types of relations could be used too. Likely candidates for this are the near-synonym and (near-)antonym relations, which are direct indicators of similarity. This would be beneficial for all parts-of-speech, but especially for verbs and adjectives.

In conclusion, it has been shown that a dependency-based word sense disambiguation algorithm, as first developed by Lin (1997), works at least as well for Dutch, using the resources readily available. Although performance on sense disambiguation tasks is hard to compare, as it depends strongly on the specifics of the test set, semantic database and correctness criteria used, performance is good for what is more or less an unsupervised method. Both in fine-grained and coarse-grained disambiguation, the algorithm performs well above a simple baseline measure. In absolute terms though, using the coarse-grained distinctions, it still only provides the correct sense for two out of three words. As such, the practical use of dependency knowledge for word sense disambiguation is likely not as a stand-alone disambiguation system. Rather, it can be utilized as a useful feature in a more advanced word sense disambiguation system that makes use of more than one information source.

References

Chen, Ping, Wei Ding, Chris Bowes, and David Brown (2009), A fully unsupervised word sense disambiguation method using dependency knowledge, *Procceedings of HLT: The 2009 Annual Conference of the NAACL*, Association for Computational Linguistics, Stroudsburg, pp. 28–36.

Dunning, Ted (1993), Accurate methods for the statistics of surprise and coincidence, *Computational Linguistics* **19** (1), pp. 61–74.

Lin, Dekang (1997), Using syntactic dependency as local context to resolve word sense ambiguity, *Proceedings of the 35th Annual Meeting of the ACL*, Association for Computational Linguistics, Stroudsburg, pp. 64–71.

van Noord, Gertjan (2006), **A**t **L**ast **P**arsing **I**s **N**ow **O**perational, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, Presses universitaires de Louvain, Leuven, pp. 20–42.

van Noord, Gertjan, Gosse Bouma, Frank van Eynde, Danil de Kok, Jelmer van der Linde, Ineke Schuurman, and Erik Tjong Kim Sang (2013), Large Scale Syntactic Annotation of Written Dutch: Lassy, *in* Spyns, Peter and Jan Odijk, editors, *Essential Speech and Language Technology for Dutch*, Springer, Berlin, chapter 9, pp. 147–164.

Vossen, P., A. Görög, F. Laan, M. van Gompel, R. Izquierdo, and Antal van den Bosch (2011), Dutchsemcor: building a semantically annotated corpus for Dutch, *Proceedings of Electronic Lexicography in the 21st century: new applications for new users*, Trojina, Institute for Applied Slovene Studies, Ljubljana, pp. 286–296.

Vossen, P., I. Maks, R. Segers, H. van der Vliet, M. Moens, K. Hofmann, and Erik Tjong Kim Sang (2013a), Cornetto: a combinatorial lexical semantic database for Dutch, *in* Spyns, Peter and Jan Odijk, editors, *Essential Speech and Language Technology for Dutch*, Springer, Berlin, chapter 10, pp. 165–184.

Vossen, P., R. Izquierdo, and A. Görög (2013b), DutchSemCor: in quest of the ideal sense tagged corpus, *Proceedings of Recent Advances in Natural Language Processing*, pp. 710–718.