UMND1: Unsupervised Word Sense Disambiguation Using Contextual Semantic Relatedness

Siddharth Patwardhan
School of Computing
University of Utah
Salt Lake City, UT 84112.
sidd@cs.utah.edu

Satanjeev Banerjee
Language Technologies Inst.
Carnegie Mellon University
Pittsburgh, PA 15217.
banerjee@cs.cmu.edu

Ted Pedersen
Dept. of Computer Science
University of Minnesota
Duluth, MN 55812.
tpederse@d.umn.edu

Abstract

In this paper we describe an unsupervised WordNet-based Word Sense Disambiguation system, which participated (as UMND1) in the SemEval-2007 Coarse-grained English Lexical Sample task. The system disambiguates a target word by using WordNet-based measures of semantic relatedness to find the sense of the word that is semantically most strongly related to the senses of the words in the context of the target word. We briefly describe this system, the configuration options used for the task, and present some analysis of the results.

1 Introduction

WordNet::SenseRelate::TargetWord[1] (Patwardhan et al., 2005; Patwardhan et al., 2003) is an unsupervised Word Sense Disambiguation (WSD) system, which is based on the hypothesis that the intended sense of an ambiguous word is related to the words in its context. For example, if the "financial institution" sense of *bank* is intended in a context, then it is highly likely the context would contain related words such as *money*, *transaction*, *interest rate*, etc. The algorithm, therefore, determines the intended sense of a word (*target word*) in a given context by measuring the relatedness of each sense of that word with the words in its context. The sense of the target word that is most related to its context is selected as the intended sense of the target word. The system uses WordNet-based measures of semantic relatedness[2] (Pedersen et al., 2004) to measure the relatedness between the different senses of the target word and the words in its context.

This system is completely unsupervised and requires no annotated data for training. The lexical database WordNet (Fellbaum, 1998) is the only resource that the system uses to measure the relatedness between words and concepts. Thus, our system is classified under the *closed track* of the task.

2 System Description

Our WSD system consists of a modular framework, which allows different algorithms for the different subtasks to be plugged into the system. We divide the disambiguation task into two primary subtasks: *context selection* and *sense selection*. The context selection module tries to select words from the context that are most likely to be indicative of the sense of the target word. The sense selection module then uses the set of selected context words to choose one of the senses of the target word as the answer.

Figure 1 shows a block schematic of the system, which takes SemEval-2007 English Lexical Sample instances as input. Each instance is a made up of a few English sentences, and one word from these sentences is marked as the target word to be disambiguated. The system processes each instance through multiple modules arranged in a sequential pipeline. The final output of the pipeline is the sense that is most appropriate for the target word in the given context.

[1] http://senserelate.sourceforge.net

[2] http://wn-similarity.sourceforge.net

Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 390–393,
Prague, June 2007. ©2007 Association for Computational Linguistics

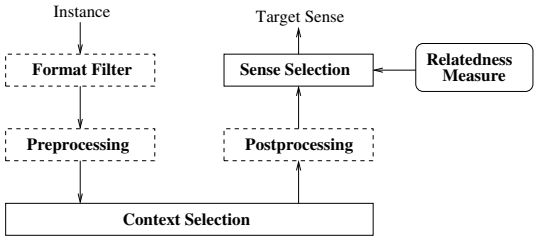

Figure 1: System Architecture

2.1 Data Preparation

The input text is first passed through a *format filter*, whose task is to parse the input XML file. This is followed by a *preprocessing* step. Each instance passed to the preprocessing stage is first segmented into words, and then all compound words are identified. Any sequence of words known to be a compound in WordNet is combined into a single entity.

2.2 Context Selection

Although each input instance consists of a large number of words, only a few of these are likely to be useful for disambiguating the target word. We use the context selection algorithm to select a subset of the context words to be used for sense selection. By removing the unimportant words, the computational complexity of the algorithm is reduced.

In this work, we use the *NearestWords* context selection algorithm. This algorithm algorithm selects $2n + 1$ content words surrounding the target word (including the target word) as the context. A stop list is used to identify closed-class non-content words. Additionally, any word not found in WordNet is also discarded. The algorithm then selects n content words before and n content words following the target word, and passes this unordered set of $2n + 1$ words to the Sense Selection module.

2.3 Sense Selection Algorithm

The sense selection module takes the set of words output by the context selection module, one of which is the target word to be disambiguated. For each of the words in this set, it retrieves a list of senses from WordNet, based on which it determines the intended sense of the target word.

The package provides two main algorithms for Sense Selection: the *local* and the *global* algorithms, as described in previous work (Banerjee and Pedersen, 2002; Patwardhan et al., 2003). In this work, we use the *local* algorithm, which is faster and was shown to perform as well as the *global* algorithm.

The *local* sense selection algorithm measures the semantic relatedness of each sense of the target word with the senses of the words in the context, and selects that sense of the target word which is most related to the context word-senses. Given the $2n + 1$ context words, the system scores each sense of the target word. Suppose the target word t has T senses, enumerated as t_1, t_2, ..., t_T. Also, suppose w_1, w_2, ..., w_{2n} are the words in the context of t, each having W_1, W_2, ..., W_{2n} senses, respectively. Then for each t_i a score is computed as

$$\text{score}(t_i) = \sum_{j=1}^{2n} \max_{k=1 \text{ to } W_j} (\text{relatedness}(t_i, w_{jk}))$$

where w_{jk} is the k^{th} sense of word w_j. The sense t_i of target word t with the highest score is selected as the intended sense of the target word.

The relatedness between two word senses is computed using a measure of semantic relatedness defined in the WordNet::Similarity software package (Pedersen et al., 2004), which is a suite of Perl modules implementing a number WordNet-based measures of semantic relatedness. For this work, we used the Context Vector measure (Patwardhan and Pedersen, 2006). The relatedness of concepts is computed based on word co-occurrence statistics derived from WordNet glosses. Given two WordNet senses, this module returns a score between 0 and 1, indicating the relatedness of the two senses.

Our system relies on WordNet as its sense inventory. However, this task used OntoNotes (Hovy et al., 2006) as the sense inventory. OntoNotes word senses are groupings of similar WordNet senses. Thus, we used the training data answer key to generate a mapping between the OntoNotes senses of the given lexical elements and their corresponding WordNet senses. We had to manually create the mappings for some of the WordNet senses, which had no corresponding OntoNotes senses. The sense selection algorithm performed all of its computations with respect to the WordNet senses, and finally the OntoNotes sense corresponding to the selected WordNet sense of the target word was output as the

answer for each instance.

3 Results and Analysis

For this task, we used the freely available WordNet::SenseRelate::TargetWord v0.10 and the WordNet::Similarity v1.04 packages. WordNet v2.1 was used as the underlying knowledge base for these. The context selection module used a window size of five (including the target word). The semantic relatedness of concepts was measured using the Context Vector measure, with configuration options as defined in previous research (Patwardhan and Pedersen, 2006). Since we always predict exactly one sense for each instance, the precision and recall values of all our experiments were always the same. Therefore, in this section we will use the name "accuracy" to mean both precision and recall.

3.1 Overall Results, and Baselines

The overall accuracy of our system on the test data is 0.538. This represents 2,609 correctly disambiguated instances, out of a total of 4,851 instances.

As baseline, we compare against the *random* algorithm where for each instance, we randomly pick one of the WordNet senses for the lexical element in that instance, and report the OntoNotes senseid it maps to as the answer. This algorithm gets an accuracy of 0.417. Thus, our algorithm gets an improvement of 12% absolute (29% relative) over this random baseline.

Additionally, we compare our algorithm against the *WordNet SenseOne* algorithm. In this algorithm, we pick the *first* sense among the WordNet senses of the lexical element in each instance, and report its corresponding OntoNotes sense as the answer for that instance. This algorithm leverages the fact that (in most cases) the WordNet senses for a particular word are listed in the database in descending order of their frequency of occurrence in the corpora from which the sense inventory was created. If the new test data has a similar distribution of senses, then this algorithm amounts to a "majority baseline". This algorithm achieves an accuracy of 0.681 which is 15% absolute (27% relative) better than our algorithm. Although this seemingly naïve algorithm outperforms our algorithm, we choose to avoid using this information in our algorithms because it represents a large amount of human supervision in the form of manual sense tagging of text, whereas our goal is to create a purely unsupervised algorithm. Additionally, our algorithms can, with little change, work with other sense inventories besides WordNet that may not have this information.

3.2 Results Disaggregated by Part of Speech

In our past experience, we have found that average disambiguation accuracy differs significantly between words of different parts of speech. For the given test data, we separately evaluated the noun and verb instances. We obtained an accuracy of 0.399 for the noun targets and 0.692 for the verb targets. Thus, we find that our algorithm performs much better on verbs than on nouns, when evaluated using the OntoNotes sense inventory. This is different from our experience with SENSEVAL data from previous years where performance on nouns was uniformly better than that on verbs. One possible reason for the better performance on verbs is that the OntoNotes sense inventory has, on average, fewer senses per verb word (4.41) than per noun word (5.71). However, additional experimentation is needed to more fully understand the difference in performance.

3.3 Results Disaggregated by Lexical Element

To gauge the accuracy of our algorithm on different words (lexical elements), we disaggregated the results by individual word. Table 1 lists the accuracy values over instances of individual verb lexical elements, and Table 2 lists the accuracy values for noun lexical elements. Our algorithm gets all instances correct for 13 verb lexical elements, and for none of the noun lexical elements. More generally, our algorithm gets an accuracy of 50% or more on 45 out of the 65 verb lexical elements, and on 15 out of the 35 noun lexical elements. For nouns, when the accuracy results are viewed in sorted order (as in Table 2), one can observe a sudden degradation of results between the accuracy of the word *system.n* – 0.443 – and the word *source.n* – 0.257. It is unclear why there is such a jump; there is no such sudden degradation in the results for the verb lexical elements.

4 Conclusions

This paper describes our system UMND1, which participated in the SemEval-2007 Coarse-grained

Word	Accuracy	Word	Accuracy
remove	1.000	purchase	1.000
negotiate	1.000	improve	1.000
hope	1.000	express	1.000
exist	1.000	estimate	1.000
describe	1.000	cause	1.000
avoid	1.000	attempt	1.000
affect	1.000	say	0.969
explain	0.944	complete	0.938
disclose	0.929	remember	0.923
allow	0.914	announce	0.900
kill	0.875	occur	0.864
do	0.836	replace	0.800
maintain	0.800	complain	0.786
believe	0.764	receive	0.750
approve	0.750	buy	0.739
produce	0.727	regard	0.714
propose	0.714	need	0.714
care	0.714	feel	0.706
recall	0.667	examine	0.667
claim	0.667	report	0.657
find	0.607	grant	0.600
work	0.558	begin	0.521
build	0.500	keep	0.463
go	0.459	contribute	0.444
rush	0.429	start	0.421
raise	0.382	end	0.381
prove	0.364	enjoy	0.357
see	0.296	set	0.262
promise	0.250	hold	0.250
lead	0.231	prepare	0.222
join	0.222	ask	0.207
come	0.186	turn	0.048
fix	0.000		

Table 1: Verb Lexical Element Accuracies

English Lexical Sample task. The system is based on WordNet::SenseRelate::TargetWord, which is a freely available unsupervised Word Sense Disambiguation software package. The system uses WordNet-based measures of semantic relatedness to select the intended sense of an ambiguous word. The system required no training data and using WordNet as its only knowledge source achieved an accuracy of 54% on the blind test set.

Acknowledgments

This research was partially supported by a National Science Foundation Early CAREER Development award (#0092784).

Word	Accuracy	Word	Accuracy
policy	0.949	people	0.904
future	0.870	drug	0.870
space	0.857	capital	0.789
effect	0.767	condition	0.765
job	0.692	bill	0.686
area	0.676	base	0.650
management	0.600	power	0.553
development	0.517	chance	0.467
exchange	0.459	order	0.456
part	0.451	president	0.446
system	0.443	source	0.257
network	0.218	state	0.208
share	0.192	rate	0.186
hour	0.167	plant	0.109
move	0.085	point	0.080
value	0.068	defense	0.048
position	0.044	carrier	0.000
authority	0.000		

Table 2: Noun Lexical Element Accuracies

References

S. Banerjee and T. Pedersen. 2002. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145, Mexico City, Mexico, February.

C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 57–60, New York, NY, June.

S. Patwardhan and T. Pedersen. 2006. Using WordNet-based Context Vectors to Estimate the Semantic Relatedness of Concepts. In *Proceedings of the EACL 2006 Workshop on Making Sense of Sense: Bringing Computational Linguistics and Psycholinguistics Together*, pages 1–8, Trento, Italy, April.

S. Patwardhan, S. Banerjee, and T. Pedersen. 2003. Using Measures of Semantic Relatedness for Word Sense Disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257, Mexico City, Mexico, February.

S. Patwardhan, T. Pedersen, and S. Banerjee. 2005. SenseRelate::TargetWord - A Generalized Framework for Word Sense Disambiguation. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (Intelligent Systems Demonstrations)*, pages 1692–1693, Pittsburgh, PA, July.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Demonstrations*, pages 38–41, Boston, MA, May.