SenseLearner: Minimally Supervised Word Sense Disambiguation for All Words in Open Text

Rada Mihalcea and Ehsanul Faruque
Department of Computer Science
University of North Texas
{rada,faruque}@cs.unt.edu

Abstract

This paper introduces SENSELEARNER – a minimally supervised sense tagger that attempts to disambiguate all content words in a text using the senses from WordNet. SENSELEARNER participated in the SENSEVAL-3 English all words task, and achieved an average accuracy of 64.6%.

1 Introduction

The task of word sense disambiguation consists of assigning the most appropriate meaning to a polysemous word within a given context. Applications such as machine translation, knowledge acquisition, common sense reasoning, and others, require knowledge about word meanings, and word sense disambiguation is considered essential for all these applications.

Most of the efforts in solving this problem were concentrated so far toward targeted supervised learning, where each sense tagged occurrence of a particular word is transformed into a feature vector, which is then used in an automatic learning process. The applicability of such supervised algorithms is however limited only to those few words for which sense tagged data is available, and their accuracy is strongly connected to the amount of labeled data available at hand.

Instead, methods that address all words in open-text have received significantly less attention. While the performance of such methods is usually exceeded by their supervised corpus-based alternatives, they have however the advantage of providing larger coverage.

In this paper, we introduce a new method for solving the semantic ambiguity of all content words in a text. The algorithm can be thought of as a minimally supervised WSD algorithm in that it uses a small data set for training purposes, and generalizes the concepts learned from the training data to disambiguate the words in the test data set. As a result, the algorithm does not need a separate classifier for each word to be disambiguated. Moreover, it does not requires thousands of occurrences of the same word to be able to disambiguate the word; in fact, it can successfully disambiguate a content word even if it did not appear in the training data.

2 Background

For some natural language processing tasks, such as part of speech tagging or named entity recognition, regardless of the approach considered, there is a consensus on what makes a successful algorithm (Resnik and Yarowsky, 1997). Instead, no such consensus has been reached yet for the task of word sense disambiguation, and previous work has considered a range of knowledge sources, such as local collocational clues, membership in a semantically or topically related word class, semantic density, etc. Other related work has been motivated by the intuition that syntactic information in a sentence contains enough information to be able to infer the semantics of words. For example, according to (Gomez, 2001), the syntax of many verbs is determined by their semantics, and thus it is possible to get the later from the former. On the other hand, (Lin, 1997) proposes a disambiguation algorithm that relies on the basic intuition that if two occurrences of the same word have identical meanings, then they should have similar local context. He then extends this assumption one step further and proposes an algorithm based on the intuition that two different words are likely to have similar meanings if they occur in an identical local context.

3 SenseLearner

Our goal is to use as little annotated data as possible, and at the same time make the algorithm general enough to be able to disambiguate all content words in a text. We are therefore using (1) SemCor (Miller et al., 1993) – a balanced, semantically annotated dataset, with all content words manually tagged by trained lexicographers – to learn a se-

SENSEVAL-3: Third International Workshop on the Evaluation of Systems
for the Semantic Analysis of Text, Barcelona, Spain, July 2004
Association for Computational Linguistics

mantic language model for the words seen in the training corpus; and (2) information drawn from WordNet (Miller, 1995), to derive semantic generalizations for those words that did not appear in the annotated corpus.

The input to the disambiguation algorithm consists of raw text. The output is a text with word meaning annotations for all open-class words.

The algorithm starts with a preprocessing stage, where the text is tokenized and annotated with parts of speech; collocations are identified using a sliding window approach, where a collocation is considered to be a sequence of words that forms a compound concept defined in WordNet; named entities are also identified at this stage.

Next, the following two main steps are applied sequentially:

1. Semantic Language Model. In the first step, a semantic language model is learned for each part of speech, starting with the annotated corpus. These models are then used to annotate words in the test corpus with their corresponding meaning. This step is applicable only to those words that appeared at least once in the training corpus.

2. Semantic Generalizations using Syntactic Dependencies and a Conceptual Network. This method is applied to those words not covered by the semantic language model. Through the semantic generalizations it makes, this second step is able to annotate words that never appeared in the training corpus.

3.1 Semantic Language Model

The role of this first module is to learn a global model for each part of speech, which can be used to disambiguate content words in any input text. Although significantly more general than models that are built individually for each word in a test corpus as in e.g. (Hoste et al., 2002) – the models can only handle words that were previously seen in the training corpus, and therefore their coverage is not 100%.

Starting with an annotated corpus formed by all annotated files in SemCor, a separate training data set is built for each part of speech. The following features are used to build the training models.

Nouns
- The first noun, verb, or adjective before the target noun, within a window of at most five words to the left, and its part of speech.

Verbs
- The first word before and the first word after the target verb, and its part of speech.

Adj
- One relying on the first noun after the target adjective, within a window of at most five words.
- A second model relying on the first word before and the first word after the target adjective, and its part of speech.

The two models for adjectives are applied individually, and then combined through voting.

For each open-class word in the training corpus (i.e. SemCor), a feature vector is built and added to the corresponding training set. The label of each such feature vector consists of the target word and the corresponding sense, represented as *word#sense*. Using this procedure, a total of 170,146 feature vectors are constructed: 86,973 vectors in the noun model, 47,838 in the verb model, and 35,335 vectors in each of the two adjective models.

To annotate new text, similar vectors are created for all content-words in the raw text. The vectors are stored in different files based on their syntactic class, and a separate learning process is run for each part-of-speech. For learning, we are using the Timbl memory based learning algorithm (Daelemans et al., 2001), which was previously found useful for the task of word sense disambiguation (Mihalcea, 2002).

Following the learning stage, each vector in the test data set – and thus each content word – is labeled with a *predicted* word and sense. If the word predicted by the learning algorithm coincides with the target word in the test feature vector, then the predicted sense is used to annotate the test instance. Otherwise, if the predicted word is different than the target word, no annotation is produced, and the word is left for annotation in a later stage.

During the evaluations on the SENSEVAL-3 English all-words data set, 1,782 words were tagged using the semantic language model, resulting in an average coverage of 85.6%.

3.2 Semantic Generalizations using Syntactic Dependencies and a Conceptual Network

Similar to (Lin, 1997), we consider the syntactic dependency of words, but we also consider the conceptual hierarchy of a word obtained through the WordNet semantic network – as a

means for generalization, capable to handle unseen words. Thus, this module can disambiguate multiple words using the same knowledge source. Moreover, the algorithm is able to disambiguate a word even if it does not appear in the training corpus. For instance, if we have a verb-object dependency pair, "drink water" in the training corpus, using the conceptual hierarchy, we will be able to successfully disambiguate the verb-object pair "take tea", even if this particular pair did not appear in the training corpus. This is done via the generalization learned from the semantic network – "drink water" allows us to infer a more general relation "take-in liquid", which in turn will help disambiguate the pair "take tea", as a specialization for "take-in liquid".

The semantic generalization algorithm is divided into two phases: training phase and test phase.

Training Phase As mentioned above, we use the annotated data provided in SemCor for training purposes. In order to combine the syntactic dependency of words and the conceptual hierarchies through WordNet hypernymy relations, the following steps are performed:

1. Remove the SGML tags from SemCor, and produce a raw file with one sentence per line.
2. Parse the sentence using the Link parser (Sleator and Temperley, 1993), and save all the dependency-pairs.
3. Add part-of-speech and sense information (as provided by SemCor) to each open word in the dependency-pairs.
4. For each noun or verb in a dependency-pair, obtain the WordNet hypernym tree of the word. We build a vector consisting of the words themselves, their part-of-speech, their WordNet sense, and a reference to all the hypernym synsets in WordNet. The reason for attaching hypernym information to each dependency pair is to allow for semantic generalizations during the learning phase.
5. For each dependency-pair, we generate positive feature vectors for the senses that appear in the training set, and negative feature vectors for all the remaining possible senses.

Test Phase After training, we can use the generalized feature vector to assign the appropriate sense to new words in a test data set. In the test phase, we complete the following steps:

1. Parse each sentences of the test file using the Link parser, and save all the dependency-pairs.
2. Start from the leftmost open word in the sentence and retrieve all the other open words it connects to.
3. For each such dependency-pair, create feature vectors for all possible combinations of senses. For example, if the first open word in the pair has two possible senses and the second one has three possible senses, this results in a total of six possible feature vectors.
4. Finally, we pass all these feature vectors to a memory based learner, Timbl (Daelemans et al., 2001), which will attempt to label each feature vector with a positive or negative label, based on information learned from the training data.

An Example Consider the following sentence from SemCor: *The Fulton County Grand Jury said Friday an investigation of Atlanta's recent primary election produced "no evidence" that any irregularities took place.* As mentioned before, the first step consists of parsing the sentence and collecting all possible dependency-pairs among words, such as subject-verb, verb-object, etc. For simplicity, let us focus on the verb-object relation between *produce* and *evidence*. We extract the proper senses of the two words from SemCor. Thus, at this point, combining the syntactic knowledge from the parser, and the semantic knowledge extracted from SemCor, we know that there is a object-verb link/relation between *produced#v#4* and *evidence#n#1*.

We now look up the hypernym tree for each of the words involved in the current dependency-pair, and create a feature vector as follows:

Os, produce#v#4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, produce#v#4, expose#v#3, show#v#4, evidence#n#1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, evidence#n#1, information#n#3, cognition#n#1, psychological_feature#n#1

where *Os* indicates an object-verb relation, and the null elements are used to pad the feature vector for a constant size of 20 elements per word.

Assuming the following sentence in the test data: *"expose meaningful information."*, we identify an object-verb relation between *expose* and *information*. Although none of the words in the pair "expose information" appear in the training corpus, by looking up the IS-A hierarchy from WordNet, we will be able to successfully disambiguate

this pair, as both "expose" and "information" appear in the feature vector (see the vector above).

4 Evaluation

The SENSELEARNER system was evaluated on the SENSEVAL-3 English all words data – a data set consisting of three texts from the Penn Treebank corpus, with a total of 2,081 annotated content words. Table 1 shows precision figures for each part-of-speech (nouns, verbs, adjectives), and contribution of each word class toward total recall.

Class	Precision	Fraction of Recall
Nouns	69.4	31.0
Verbs	56.1	20.2
Adjectives	71.6	12.2
Total	64.6	64.6

Table 1: SENSELEARNER results in the SENSEVAL-3 English all words task

The average precision of 64.6% compares favorably with the "most frequent sense" baseline, which was computed at 60.9%. Not surprisingly, the verbs seem to be the most difficult word class, which is most likely explained by the large number of senses defined in WordNet for this part of speech.

5 Conclusion

In this paper, we proposed and evaluated a new algorithm for minimally supervised word-sense disambiguation that attempts to disambiguate all content words in a text using the senses from WordNet. The algorithm was implemented in a system called SENSELEARNER, which participated in the SENSEVAL-3 English all words task and obtained an average accuracy of 64.6% – a significant improvement over the most frequent sense baseline of 60.9%.

Acknowledgments

This work was partially supported by a National Science Foundation grant IIS-0336793.

References

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2001. Timbl: Tilburg memory based learner, version 4.0, reference guide. Technical report, University of Antwerp.

F. Gomez. 2001. An algorithm for aspects of semantic interpretation using an enhanced Wordnet. In *Proceedings of the North American Association for Computational Linguistics (NAACL 2001)*, Pittsburgh, PA.

V. Hoste, W. Daelemans, I. Hendrickx, and A. van den Bosch. 2002. Evaluating the results of a memory-based word-expert approach to unrestricted word sense disambiguation. In *Proceedings of the ACL Workshop on "Word Sense Disambiguatuion: Recent Successes and Future Directions"*, Philadelphia, July.

D. Lin. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the Association for Computational Linguistics*, Madrid, Spain.

R. Mihalcea. 2002. Instance based learning with automatic feature selection applied to Word Sense Disambiguation. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan, August.

G. Miller, C. Leacock, T. Randee, and R. Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, New Jersey.

G. Miller. 1995. Wordnet: A lexical database. *Communication of the ACM*, 38(11):39–41.

P. Resnik and D. Yarowsky. 1997. A perspective on word sense disambiguation methods and their evaluation. In *Proceedings of ACL Siglex Workshop on Tagging Text with Lexical Semantics, Why, What and How?*, Washington DC, April.

D. Sleator and D. Temperley. 1993. Parsing English with a Link grammar. In *Third International Workshop on Parsing Technologies*.