SenseRelate::TargetWord – A Generalized Framework for Word Sense Disambiguation

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 15213
satanjeev@cmu.edu

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

Abstract

Many words in natural language have different meanings when used in different contexts. *SenseRelate::TargetWord* is a Perl package that disambiguates a target word in context by finding the sense that is most related to its neighbors according to a *WordNet::Similarity* measure of relatedness.

Introduction

Word Sense Disambiguation is the task of identifying the intended meaning of a given *target word* from the context in which it is used. (Lesk 1986) performed disambiguation by counting the number of overlaps between the dictionary definitions (i.e., glosses) of the target word and those of the neighboring words in the context. (Banerjee & Pedersen 2002) extended this method of disambiguation by expanding the glosses of words to include glosses of related words, according to the structure of WordNet. In subsequent work, (Patwardhan, Banerjee, & Pedersen 2003) and (Banerjee & Pedersen 2003) proposed that measuring gloss overlaps is just one way of determining *semantic relatedness*, and that word sense disambiguation can be performed by finding the most related sense of a target word to its surrounding context, using any of the measures of relatedness provided in the freely available package *WordNet::Similarity*(Pedersen, Patwardhan, & Michelizzi 2004).

These ideas are all implemented in the freely available software package *SenseRelate::TargetWord*.

The Framework

The package has a highly modular architecture. The disambiguation process is divided into a number of smaller subtasks, implemented as separate modules. Figure 1 depicts the architecture of the system and shows the various subtasks. Each of the sequential sub-tasks or *stages* accept data from a previous stage, perform a transformation on the data, and then pass on the processed data structures to the next stage in the pipeline. A user can create her own modules to perform any of these sub-tasks as long as the modules adhere to the protocol laid down by the package.

Format Filter: The filter takes as input file(s) annotated in the SENSEVAL-2 lexical sample format. A file in this XML–based format includes a number of *instances*, each one made up of 2 to 3 lines of text where a single target word is designated with an XML tag. The filter parses the input file to build data structures that represent the instances to be disambiguated, which includes a single target word and the surrounding words that define the context.

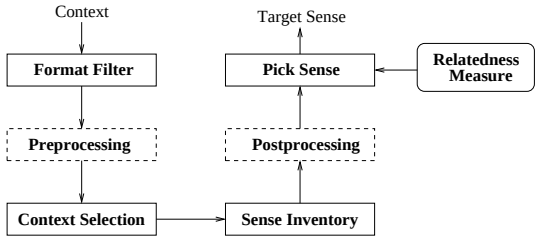

Figure 1: Our Framework for *Word Sense Disambiguation*

Preprocessing: *SenseRelate::TargetWord* expects zero or more text preprocessing modules, each of which perform a transformation on the input words. For example, the *Compound Detection Module* identifies sequences of tokens that form compound words. Multiple preprocessing modules can be chained together, the output of one connected to the input of the next, to form a single preprocessing stage. For example, a part of speech tagging module could be added after compound detection.

Context Selection: The standard context selection module that is provided takes as input the target word and all the words in its surrounding context. The module then selects n words (including the target word) from this list and sends this list on to the next module for disambiguation. Ideally these words would be the most indicative of the correct sense of the target word.

In the simplest case, we could select $n-1$ words nearest to the target word. However, we could incorporate some intelligence into the context selection. For example, we could select the $n-1$ nearest nouns having a high term frequency to document frequency ratio.

Sense Inventory: After having reduced the context to n words, the *Sense Inventory* stage determines the possible senses of each of the n words. This list can be obtained from a dictionary, such as WordNet.

In our system, this module first decides the base (unin-

flected) form of each of the n words. It then retrieves all the senses for each word from the sense inventory. We use WordNet for our sense inventory.

Postprocessing: Some optional processing can be performed on the data structures generated by the Sense Inventory module. This would include tasks such as *sense pruning*, which is the process of removing some senses from the inventory, based on simple heuristics, algorithms or options. For example, the user may decide to preclude all verb senses of the target word from further consideration in the disambiguation process.

Sense Selection: The disambiguation module takes the lists of senses of the target word and those of the context words and uses this information to pick one sense of the target word as the answer. Many different algorithms could be used to do this. We have modules *Local* and *Global* that (in different ways) determine the relatedness of each of the senses of the target word with those of the context words, and pick the most related sense as the answer. These are described in greater detail by (Banerjee & Pedersen 2002).

Using SenseRelate::TargetWord

SenseRelate::TargetWord can be used via the command-line interface (*disamb.pl*), a graphical interface, as well as a programming API.

The command-line interface *disamb.pl* takes as input a SENSEVAL-2 formatted lexical sample file. The program disambiguates the marked up word in each instance and prints to screen the instance ID, along with the disambiguated sense of the target word. Many command line options are available to control the disambiguation process.

SenseRelate::TargetWord is distributed as a Perl package. It is programmed in object-oriented Perl as a group of Perl classes. Objects of these classes can be instantiated in user programs, and methods can be called on these objects.

We are currently developing a graphical interface for the package in order to conveniently access the disambiguation modules. The GUI is being written in Gtk-Perl – a Perl API to the Gtk toolkit. Unlike the command line interface, the graphical interface will not be tied to any input file format. The interface will allow the user to input text, and to select the word to disambiguate.

Related Work

One of the first approaches to Word Sense Disambiguation that used content from a dictionary was that of (Lesk 1986), which treated every dictionary definition of a concept as a bag of words. To identify the intended sense of the target word, the Lesk algorithm would determine the number of word overlaps between the definitions of each of the meanings of the target word, and those of the context words. The meaning of the target word with maximum definition overlap with the context words was selected as the intended sense.

(Wilks *et al.* 1993) developed a context vector approach for performing word sense disambiguation. Their algorithm built co-occurrence vectors from dictionary definitions using Longman's Dictionary of Contemporary English (LDOCE). They then determined the extent of overlap between the sum of the vectors of the words in the context and the sum of the vectors of the words in each of the definitions (of the target word). For vectors, the extent of overlap is defined as the dot product of the vectors. The meaning of the target word that had the maximum overlap was selected as the answer.

More recently, (McCarthy *et al.* 2004) present a method that performs disambiguation by determing the most frequent sense of a word in a particular domain. This is based on measuring the relatedness of the different possible senses of a target word (using *WordNet::Similarity*) to a set of words associated with a particular domain that have been identified using distributional methods. The relatedness scores between a target word and the members of this set are scaled by the distributional similarity score.

Acknowledgements

This research is partially supported by a National Science Foundation Faculty Early CAREER Development Award (#0092784).

SenseRelate::TargetWord is written in Perl and is freely distributed under the Gnu Public License. It is available via SourceForge, an Open Source development platform[1], and the Comprehensive Perl Archive Network (CPAN)[2].

References

Banerjee, S., and Pedersen, T. 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*.

Banerjee, S., and Pedersen, T. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence (IJCAI-03)*.

Lesk, M. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone. In *Proceedings of SIGDOC '86*.

McCarthy, D.; Koeling, R.; Weeds, J.; and Carroll, J. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, 279–286.

Patwardhan, S.; Banerjee, S.; and Pedersen, T. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics (CICLING-03)*.

Pedersen, T.; Patwardhan, S.; and Michelizzi, J. 2004. Wordnet::Similarity - measuring the relatedness of concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 1024–1025.

Wilks, Y.; Fass, D.; Guo, C.; McDonald, J.; Plate, T.; and Slator, B. 1993. Providing machine tractable dictionary tools. In Pustejovsky, J., ed., *Semantics and the Lexicon*. Dordrecht and Boston: Kluwer Academic Press.

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

[2] http://search.cpan.org/dist/WordNet-SenseRelate-TargetWord