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

We have previously introduced a method of word sense disambiguation that computes the intended sense of a target word, using WordNet-based measures of semantic relatedness (Patwardhan et al., 2003). *SenseRelate::TargetWord* is a Perl package that implements this algorithm. The disambiguation process is carried out by selecting that sense of the target word which is most related to the context words. Relatedness between word senses is measured using the *WordNet::Similarity* Perl modules.

1 Introduction

Many words have different meanings when used in different contexts. *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 and 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 (Fellbaum, 1998). In subsequent work, (Patwardhan et al., 2003) and (Banerjee and Pedersen, 2003) proposed that measuring gloss overalps 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 a wide variety of measures of relatedness.

SenseRelate::TargetWord is a Perl package that implements these ideas, and is able to disambiguate a target word in context by finding the sense that is most related to its neighbors according to a specified measure. A user of this package is able to make a variety of choices for *text preprocessing options*, *context selection*, *relatedness measure selection* and the selection of an *algorithm* for computing the overall relatedness between each sense of the target word and words in the surrounding context. The user can customize each of these choices to fit the needs of her specific disambiguation task. Further, the various sub-tasks in the package are implemented in a modular fashion, allowing the user to easily replace a module with her own module if needed.

The following sections describe the generalized framework for Word Sense Disambiguation, the architecture and usage of *SenseRelate::TargetWord*, and a description of the user interfaces (command line and GUI).

2 The Framework

The package has a highly modular architecture. The disambiguation process is divided into a number of smaller sub-tasks, each of which is represented by a separate module. Each of the sequential sub-tasks or *stages* accepts data from a previous stage, performs a transformation on the data, and then passes on the processed data structures to the next stage in the pipeline. We have created a protocol that defines the structure and format of the data that is passed between the stages. The user can create her own

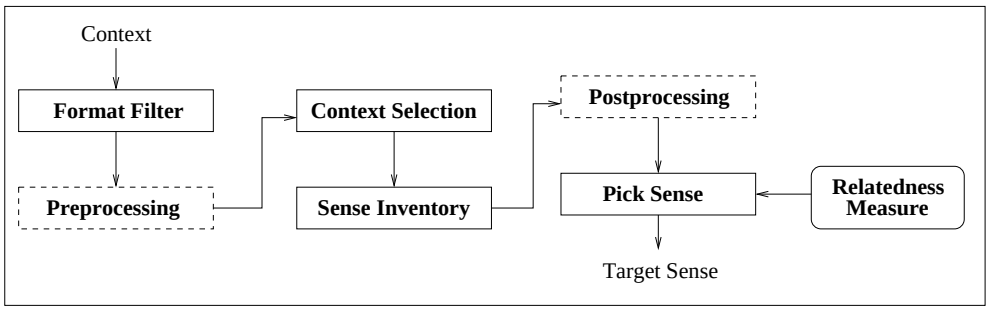

Figure 1: A generalized framework for *Word Sense Disambiguation*.

modules to perform any of these sub-tasks as long as the modules adhere to the protocol laid down by the package.

Figure 1 projects an overview of the architecture of the system and shows the various sub-tasks that need to be performed to carry out word sense disambiguation. The sub-tasks in the dotted boxes are optional. Further, each of the sub-tasks can be performed in a number of different ways, implying that the package can be customized in a large number of ways to suit different disambiguation needs.

2.1 Format Filter

The filter takes as input file(s) annotated in the SENSEVAL-2 lexical sample format, which is an XML–based format that has been used for both the SENSEVAL-2 and SENSEVAL-3 exercises. A file in this 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.

2.2 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 that are known as concepts to WordNet (such as "New York City"). In order to ensure that compounds are treated as a single unit, the package replaces them in the instance with the corresponding underscore–connected form ("New_York_City").

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.

2.3 Context Selection

Disambiguation is performed by finding the sense of the target word that is most related to the words in its surrounding context. The package allows for various methods of determining what exactly the surrounding context should consist of. In the current implementation, the context selection module uses an n word window around the target word as context. The window includes the target word, and extends to both the left and right. The module selects the $n-1$ words that are located closest to the target word, and sends these words (and the target) on to the next module for disambiguation. Note that these words must all be known to WordNet, and should not include any stop–words.

However, not all words in the surrounding context are indicative of the correct sense of the target word. An intelligent selection of the context words used in the disambiguation process could potentially yield much better results and generate a solution faster than if all the nearby words were used. For example, we could instead select the nouns from the window of context that have a high term–frequency to document–frequency ratio. Or, we could identify lexical chains in the surrounding context, and only include those words that are found in chains that include the target word.

2.4 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. A thesaurus could also be used for the purpose. Note however, that the subsequent modules in the pipeline should be aware of the codes assigned to the word senses.

In our system, this module first decides the base (uninflected) 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.

2.5 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.

2.6 Identifying the Sense

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 and Pedersen, 2002), but in general the Local method compares the target word to its neighbors in a pair-wise fashion, while the Global method carries out an exhaustive comparison between all the senses of the target word and all the senses of the neighbors.

3 Using SenseRelate::TargetWord

SenseRelate::TargetWord can be used via the command-line interface provided by the utility program called *disamb.pl*. It provides a rich variety of options for controlling the process of disambiguation. Or, it can be embedded into Perl programs, by including it as a module and calling its various methods. Finally, there is a graphical interface to the package that allows a user to highlight a word in context to be disambiguated.

3.1 Command Line

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.

Command line options are available to control the disambiguation process. For example, a user can specify which relatedness measure they would like to use, whether disambiguation should be carried out using Local or Global methods, how large a window of context around the target word is to be used, and whether or not all the parts of speech of a word should be considered.

3.2 Programming Interface

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. The package requires that the Perl interface to WordNet, *WordNet::QueryData*[1] be installed on the system. The disambiguation algorithms also require that the semantic relatedness measures *WordNet::Similarity* (Pedersen et al., 2004) be installed.

3.3 Graphical User Interface

We have developed a graphical interface for the package in order to conveniently access the disambiguation modules. The GUI is written in Gtk-Perl – a Perl API to the Gtk toolkit. Unlike the command line interface, the graphical interface is not tied to any input file format. The interface allows the user to input text, and to select the word to disambiguate. It also provides the user with numerous configuration options corresponding to the various customizations described above.

[1] http://search.cpan.org/dist/WordNet-QueryData

4 Related Work

There is a long history of work in Word Sense Disambiguation that uses Machine Readable Dictionaries, and are highly related to our approach.

One of the first approaches 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.

5 Availability

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[2], and the Comprehensive Perl Archive Network (CPAN)[3].

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

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

6 Acknowledgements

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

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*, Mexico City, February.

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

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

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

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 279–286, Barcelona, Spain, July.

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 (CICLING-03)*, Mexico City, Mexico, February.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::Similarity - Measuring the Relatedness of Concepts. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Demonstration Papers*, pages 38–41, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

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