CS65: An NLTK Package for Lexical-Chain Based Word Sense Disambiguation

Malcolm Augat
Swarthmore College
500 College Avenue
Swarthmore, PA 19081
maugat1@swarthmore.edu

Margaret Ladlow
Swarthmore College
500 College Avenue
Swarthmore, PA 19081
mladlow1@swarthmore.edu

Abstract

This study attempts to perform word sense disambiguation (WSD) using lexical chains, sets of semantically related words. The system uses the semantic relations defined in WordNet to determine whether two words are similar. We assume that words will be used in one sense per discourse. To determine the sense of the word, we find the sense that is most related to other words in the text. Our WSD accuracy on SEMCOR (Mihalcea, 1998) is lower than selecting the most frequent word sense for each word, but higher than that achieved in previous research (Galley and McKeown, 2003). We obtained our highest accuracy of 76.01% when using the Lancaster stemmer on words that were not initially found in WordNet and only linking words which shared a synset.

1 Introduction

The goal of this study is to develop a lexical chain based word sense disambiguation (WSD) system compatible with the Natural Language Toolkit (NLTK) (Bird and Loper, 2004). Because NLTK is a set of natural language processing tools for Python, all our code was written in said language. The resulting algorithm performs WSD using a one sense per discourse assumption.

Lexical chains are chains of words that bear some kind of semantic relationship to one another. To qualify for membership in a chain, a word must be related in some way to some other word in the chain. In the past, lexical chains have been used for summarizing the main points of a discourse (Barzilay and Elhadad, 1997; Silber and McCoy, 2003), and for word sense disambiguation (Morris and Hirst, 1991; Hirst and St-Onge, 1998; Galley and McKeown, 2003). To determine whether and in what way words are related, our system, like those of our academic predecessors, uses WordNet (Fellbaum, 1998) as a knowledge source. While WordNet has separate dictionaries for nouns, verbs, adjectives, and adverbs, the noun dictionary has the most well defined hierarchy. Because hierarchy in WordNet is very important to our WSD system, we decided to only analyze nouns.

Each word in WordNet has at least one set of synonyms, called a synset. When there are multiple synsets, each corresponds to the synonyms for a different meaning of the word. Each synset of a word is indexed by a sense number which together with the word, uniquely specifies the synset. The synsets can also be uniquely specified by their offset number. In addition, synsets in WordNet link to other synsets which represent hypernym and hyponym meanings of the words in the synset. This gives WordNet a tree structure and allows distances between two different synsets, each representing a different word, to be computed.

The WordNet distance between two synsets is the number of hypernym/hyponym links that must be followed to get from one synset to the other. For example, Dog and domestic dog have a WordNet distance of zero because they are in the same synset.

Essentially, our algorithm creates edges between all related words seen in a corpus and weights the edges based on the WordNet distance, with closer distances yielding more highly weighted edges.

When we have finished connecting all related words in a document, the algorithm finds the sense of each word with the highest sum of relationship scores. We consider that sense to be the one sense of the word used in the discourse.

The format of this paper is as follows: In Section 2 we discuss other work using lexical chains. Section 3 describes the method we chose for implementing our algorithm. In Section 4 we explore our results. Finally, we discuss possible improvements to the algorithm in Section 5.

2 Related Work

Morris and Hirst(1991) first discussed the idea of using lexical chains to determine relations between words in a discourse. They suggested that a thesaurus could be used to determine whether or not a there exists a semantic link between two words and laid out rules for making such a decision. Furthermore, they demonstrated that lexical chains might describe the structure of a text. They were unable to implement their algorithm because there was no on-line thesaurus for them to use.

The idea of using WordNet as a thesaurus for creating lexical chains was introduced by Hirst and St. Onge(1998). They defined extra-strong, medium-strong, and strong relationships between word senses. Like our system, relationship strength is dependent on WordNet distance. To make chains, they consider nouns one by one, adding them to lexical chains with related nouns or in chains on their own if no such related nouns are found. When a noun is added to a chain, all synsets in the chain that are not related to synsets of the added noun are pruned immediately.

This immediate removal of all other synsets might elminate senses of nouns that are actually more strongly related to later words in the document. Barzilay and Elhadad (1997) improved on this problem by building graphs of all the potential connections between synsets and only pruning when memory became full. However, this implementation sacrificed the linear runtime of previous approaches.

In light of this defect, Silber and McCoy(2003) developed an algorithm for text summarization using lexical chaining that implicitly represents all possible connections between words in the text. The number of connections a word occurence can have is bounded by the constant size of WordNet, allowing their algorithm to be linear with respect to the number of words in the text.

Galley and McKeown(2003) came up with a modification of Silber and McCoy's algorithm specifically for use in WSD. Their algorithm maintains the linear runtime of Silber and McCoy but links together word senses instead of word occurences. This detail causes them to rely more heavily on the assumption of one sense per discourse, an assumption which seems to be valid as their WSD accuracy of 62.09% is 7.61% higher than that achieved by the former algorithm at the same task.

3 Algorithms and Methods

The algorithm we use for lexical chaining is based on the linear time algorithm of Galley and McKeown (2003) with some modifications. Rather than using an array indexed by senses of WordNet as Galley and McKeown did, we instead used Python's built in hashmap implementation to connect words to lexical chains. This allows us to more efficiently represent the set of lexical chains for a text since for most texts the number of distinct words in the text is far smaller than the size of WordNet. This modification does mean that out algorithm, unlike Galley and McKeown's is not guaranteed to be linear unless hashing is perfect.

Additionally, our implementation is more general with regards to WordNet distances. Galley and McKeown only consider words within one step of each other in WordNet and sibling words. We allow any distances to be considered, but only consider distances from zero to two here. For example, with a distance of two each sense's parents, children, grandparents, grandchildren, and siblings are looked at for chaining.

Our code consists of four classes and a free function. The LexNode class is fairly straightforward. It stores the word, the word's sense number, the chains in which the word appears, and links to related LexNodes. Links between LexNodes are represented by the LexLink class, which stores pointers to two LexNodes. Instances of the class also store the text and WordNet distance between LexNodes. LexScoring stores weights defined by the user, though a default is included in our system. LexLinks are assigned a score based both on text distance and WordNet distance, and the LexScoring object is used to determine the weights on each LexLink.

The most important class is the MetaChain class, which stores representations of all possible chains. The most notable function in MetaChain is addWord, where we add a word to all of its possible lexical chains. When we see a word that we have previously encountered, we simply check to see if the text position of this word is closer to any related words than the previous text distance stored for the word. If a word is not in WordNet, we assign it a default sense number. By default, this value is None, but it can be changed by the user. If we have not seen the word before, and if the word can be found in WordNet, then for every synset of the word, we make a new LexNode. We then make LexLinks between the LexNode and everything within its WordNet distance.

In order to evaluate the efficacy of our algorithm we tested it against the semantically labeled Brown corpus of SEMCOR (1998). For each Brown text we got the WordNet sense number for each word and then compared this to the given sense number in SEMCOR to get an overall accuracy of our word sense disambiguation.

4 Results

The accuracy of our system on the Brown corpus of SEMCOR can be seen in Table 1, and detailed results of the accuracies on words with different degrees of polysemy can be seen in Figure 1. We tested our algorithm using lexical chaining of words with WordNet distances between zero (only linking word senses which share synsets) and two (linking word senses as far apart as siblings or grandparents/children) and found that resticting lexical chaining to smaller WordNet distances increased accuracy. Our weighting scheme for all cases was the same as that presented in (Galley and McKeown, 2003). We use WSD of each word as its most common sense in WordNet as our baseline accuracy. Note that this number is somewhat deceptive and circular since WordNet sense rankings are derived from the SEMCOR corpus on which we evaluated accuracy, possible inflating the baseline.

Some number of the successfully matched words are due to the fact that for words not in WordNet or for words with only one sense there was little chance of failure since these words are tagged with sense number one in SEMCOR. As these are not truly successes in word sense disambiguation we examined accuracy on WordNet distances of zero and one if we excluded these words, and again found that smaller WordNet distances increased accuracy.

Because we suspected that using small WordNet distances on smaller text might decrease accuracy due to sparse data issues we evaluated the accuracy of WordNet distances of zero and one at the task of disambiguating only the first four, twenty, and one-hundred sentences of each text (results can again be seen in Table 1). Despite our expectations we still found that smaller WordNet distances resulted in higher accuracy. This trend may be due to the higher accuracy of the baseline. When a word cannot be disambiguated (if it is not linked to any other words in a lexical chain, for instance) then the highest frequency sense is chosen for it. With the shorter texts fewer of the words may have been linked, and the very accurate baseline chosen.

Finally, we tested to see the effect of using a stemmer on unknown words. In these cases, when we encountered a noun which wasn't in WordNet we tried to find its stem and used that instead. We used three of the stemmers from the nltk.stem package: the WordNet stemmer, Lancaster stemmer, and Porter stemmer. All three we evaluated using WordNet distances of zero, and of the three the Lancaster stemmer performed the best, and the Lancaster and Porter stemmers resulted in a slight increase in accuracy over the accuracy seen when not using the stemmer. Results can once again be seen in Table 1.

5 Future Work

Our algorithm assumes one sense per word per discourse. We believe that better accuracy could be achieved if our algorithm were permitted to return one sense per word instance. To do this, we think that an algorithm would have to consider each word

WSD method	WN dist	Accuracy
Baseline	-	0.7856
Baseline, no monosemy	-	0.7100
Lex chain	0	0.7368
Lex chain	1	0.6903
Lex chain	2	0.6600
No monosemy	0	0.6234
No monosemy	1	0.5410
4 sentences	0	0.7766
20 sentences	0	0.7603
100 sentences	0	0.7375
4 sentences	1	0.7612
20 sentences	1	0.7239
100 sentences	1	0.6916
WordNet stemmer	0	0.7208
Lancaster stemmer	0	0.7601
Porter stemmer	0	0.7469

Table 1: Word sense disambiguation accuracies of the methods tested.

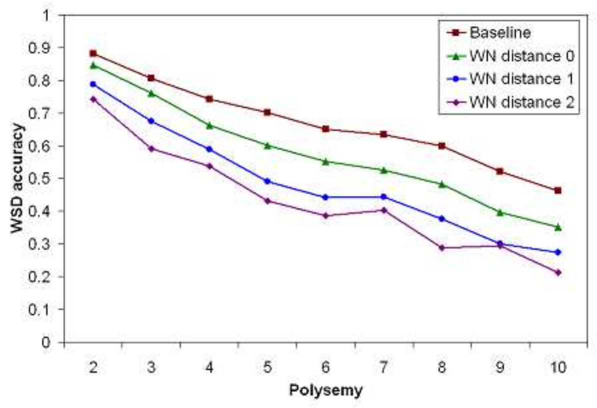

Figure 1: Word sense disambiguation accuracy for words with varying numbers of meanings (polysemies). Accuracies were calculated using the baseline most common sense for each word, and our lexical chaining algorithm with WordNet distances between zero and two. Computed by comparison with texts from the Brown corpus in SEMCOR.

occurance as a separate LexNode-like object.

Although we used only nouns for WSD in this project, we believe that interesting conclusions could be drawn by considering other parts of speech. Our algorithm is very dependent on the tree-like structure of the WordNet noun dictionary. Because the structure of the WordNet adjective, verb, and adverb dictionaries are less strictly tree-like, we believe that a similar algorithm would have to use different types of semantic relationships for WSD.

Allowing different weights specifically for different WordNet relationships rather than simply different WordNet distances could yield better accuracy. For example, it might be interesting to weight siblings differently from grandparent relationships. Now, they are both at WordNet distance 2, and are weighted equivalently.

6 Conclusion

In this project we built a lexical chain based method for word sense disambiguation based on the algorithm of Galley and McKeown (2003) with some modifications and intended for incorporation into the Natural Language ToolKit for Python. We evaluated our algorithm on the semantically labeled corpus SEMCOR, and achieved our highest accuracy of 76.01% using NLTK's Lancaster stemmer and only linking very closely related words which shared a synset.

Although the accuracy of even our best methods was under that of our baseline's 78.56% accuracy (a number which has some degree of suspicion associated with it's validity), these methods do seem to show some promise. Our overall accuracy is higher than that of Galley and McKeown (62.09% overall accuracy) which they obtained from testing on a subset of the same SEMCOR corpora.

References

Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. *ACL 1997 Workshop on Intelligent Scalable Text Summarization*.

Steven Bird and Edward Loper. 2004. Nltk: The natural language toolkit. *Proc. Association for Computational Linguistics*.

editor Fellbaum, C. 1998. Wordnet: an electronic lexical database.

Michel Galley and Kathleen McKeown. 2003. Improving word sense disambiguation in lexical chaining. *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*.

Graeme Hirst and David. St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An Electronic-Lexical Database*.

Rada Mihalcea. 1998. Semcor: Semantically tagged corpus.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. In *Computational Linguistics*.

H. Gregory Silber and Kathleen F. McCoy. 2003. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. In *Computational Linguistics*.