Word Sense Disambiguation: Pipelining with Supervised LSA

Kiran Vodrahalli, Evelyn Ding, Albert Lee

May 8, 2014

1 Introduction

1.1 Problem Statement

Word sense disambiguation is the process of identifying the meaning of a word in its context when an individual word may take on many meanings. It remains one of the oldest open problems in computational linguistics and is widely considered to be AI-Complete, or as hard as the problem of making computers as intelligent as people [2]. Being able to accurately disambiguate word senses would greatly assist Machine Translation, semantic analysis, and would have significant impact in computational linguistics. However, fine-grained senses in which a word has multiple closely related meanings remains one of the largest challenges in word sense disambiguation. Consider the following example of disambiguating the sense of the word *art* from the Senseval-2 competition held in 2001 [1]:

1. "Paintings, drawings and sculpture from every period of *art* during the last 350 years will be on display ranging from a Tudor portrait to contemporary British art."
2. "Through casual meetings at cafe's, the artists drew together to form a movement in protest against the waste of war, against nationalism and against everything pompous, conventional or boring in the *art* of the Western world."

In the first example, *art* refers to *the creation of beautiful or significant things* whereas in the second example, it refers to *the products of human creativity; works of art collectively*. Such subtle distinctions are part of what makes word sense disambiguation a challenging problem that remains open to this day.

2 Literature Survey

The past word sense disambiguation methods fall into three main categories: knowledge-based, unsupervised corpus-based, and supervised corpus-based methods [9].

2.1 Knowledge-based

One widely used knowledge-based algorithm is Lesk's algorithm [2]. The algorithm is as follows:

```
Given that there are two words W1 and W2 with multiple word senses:
for each sense i in W1
    for each sense j in W2
        calculate overlap of i and j in dictionary definitions
Maximize overlap i and j to determine the respective word senses of W1 and W2.
```

Another knowledge-based algorithm is semantic similarity. In order to calculate semantic similarity, calculate the path between words in Wordnet for each possible word sense of the ambiguous word with other words in the context. To select the correct word sense, minimize the total semantic path. The context of the word can vary from one sentence to the length of the document. Generally, only nouns are used to provide context.

2.2 Unsupervised Corpus-based

One of the challenges with word sense disambiguation is the scarcity of accurately sense-tagged text, which such datasets usually needing to be manually built by humans. Unsupervied approaches to word sense disambiguation attempt to overcome the bottleneck of acquiring sense-tagged text by operating on untagged corpora and clustering based on either context or word. Clustering by Committee (CBC) is an example of such an approach, formulated by Lin and Pantel in 2002 [2], in which given a target word, clusters of similar words will be output. CBC is a three step algorithm starting first with the construction of a co-occurrence matrix containing measures of the similarity between each word, followed by clustering the most similar elements from this co-occrrence matrix using average link clustering to form the committees, with each target word then being assigned to its most similar committee in the final step.

2.3 Supervised Corpus-based

The supervised corpus-based methods use an annotated corpus to build a model, and makes predictions on previous word senses based on this model. For example, for the definition to know, it could mean:

1) be aware of piece of information

Usages: I want to know who is winning. I know that the president lied.

2) know a person

She doesn't know the composer.

Usages: Do you know my sister?

The model uses the context of how a word usually appears and creates bigram/trigram models on surrounding words. These models are applied to determine word sense in other instances.

2.4 LSA – Past Efforts

LSA was one of the first approaches to make use of the co-occurrence model for word meaning. The idea, as implemented in the famous 1998 paper by Landauer [12], is that given a set of corpuses, we build a term-document matrix where words are on the rows and documents are on the columns, and the frequencies are in the corresponding cells. Then we apply SVD to this matrix. This approach is essentially unsupervised because it relies on clustering (word co-occurrence matrix) to derive similarities between documents and words.

Originally, the LSA approach was primarily intended to remove uninformative singular values, and then reconstruct the term-document matrix (Landauer et al 1998). The idea there was to compare documents to see how similar they were to each other with noise removed. Part of their motivation was to see if co-occurrence representations was in fact how humans represented meaning themselves.

(Katz et al) [6] first applied LSA to WSD in an unsupervised approach by folding new documents with ambiguous word into the semantic space created by the SVD of the term-document matrix, and then used cosine-based clustering. We do a supervised implementation of this approach.

It is worth noting that other similar approaches exist in recent years that are probably more powerful (but more difficult to implement). We detail them in the next sections.

2.4.1 Probabilistic LSA

Formulated as early as 1999 in a paper by Hofmann [10], this approach is based on mixture decomposition from latent class model. It uses a probabilistic model known as an aspect model, which is a latent variable model for co-occurrence data. It introduces a conditional dependence assumption, namely that the document random variable and the word random variable are independent conditioned on the state of the associated latent variable.

The Expectation Maximization (EM) algorithm is used for maximum likelihood estimation, as is standard procedure for latent variable models. This paper also introduces the notion of continuous latent space, produced by smoothing the discrete distribution due to the representation as a multinomial distribution consisting of a convex combination of factors. This also results in dimension reduction, and we end up with a probablistic latent semantic space.

The crucial difference between this approach and standard LSA is that LSA, the objective function to determine the optimal decomposition is results from the L_2 or Frobenius norm, which brings along with it an internal assumption of Gaussian noise. The PLSA model intends to maximize the predictive power of the model instead, which is related to minimizing cross-entropy, or Kullback-Leibler divergence.

LSA does not define a properly normalized probability distribution, and this is one of its weaknesses.

2.4.2 Non-Negative Matrix Factorization

Another alternative to using SVD is NNMF, which is a matrix factoring technique introduced by Lee and Seung (2000). The primary advantages of this approach again relate to minimizing the Kullback-Leibler divergence instead of Euclidean distance. Again, minimizing Kullback-Leibler divergence is better since minimizing Euclidean distance makes the normal distribution assumption for language – and language is not Gaussian. It is also useful for extraction of semantic dimension since it turns out to have no negative relations encoded (all matrix values are positive).

Van de Cruys et al (2011) [11] applies this approach to the SEMEVAL-2010 dataset. The rest of the approach is analogous to SVD-LSA, computing the centroid of candidate vectors and comparing via Kullback-Leibler divergence to the target vector to determine word sense.

3 Data

We used the Senseval-2 dataset from 2001 [1], which consists of 73 words with multiple senses to be disambiguated. In total, there were 8611 training instances labeled with the WordNet sense key of the target word to be disambiguated, and 4328 test instances. All the data was provided in XML format. After parsing all training instances, we used only the sentence containing each target word for our POS bigram models and some of our dictionary-based methods, but the entire paragraph context for our LSA-based method. In addition to Senseval-2 dataset, we used the Brown corpus in our LSA-based method.

4 Algorithms

Our approach was essentially a pipeline of different sorts of models. The first stage consisted of part-of-speech tagging and bigram models, which was used for the subset of data that had high predictability. Originally, we implemented a step that included knowledge-base approaches like semantic similarity, but later discarded it from the pipeline because its accuracy was not very good.

For the rest of the data, we implemented a Supervised LSA approach, which was to an extent a novel approach.

We detail the methods in the following sections.

4.1 Part-of-Speech

We constructed two POS bigram models to try and perform coarse-filtering on the data by identifying the senses we could predict with high probability using a part-of-speech model before propogating the data instances further down the pipeline. We constructed one bigram model consisting of the POS bigrams containing the word coming before the target word and the target word, and another bigram model consisting of the POS bigrams containing the target word and word following the target word. The POS tags were generated by running NLTK's POS-tagger [3] on the sentence containing the target word in each of our training data instances. On an input test instance, we would tag the sentence of interest, extract the two bigrams containing the target word, check if either bigram could be found in either of our bigram models, and output a predicted sense key if our predicted probability of the sense key given that bigram exceeded 80%. To avoid skewed probabilities from having only one training data instance with that bigram, but also to ensure sufficient coverage for the already sparse data set, we also set a frequency count threshold of three such that we would only output our predicted sense key if we had seen more than three instances of this particular bigram in our training data and we predicted a probability of over 80%.

4.2 Semantic Knowledge base

We attempted a couple of methods that were dictionary-based, but they yielded accuracy rates of $< 30\%$, so we discarded them from the pipeline.

We used Lesk's algorithm on the Brown corpus, with an implementation that was based upon a previous implementation in python. [5, 7] We used a basic form of Lesk's algorithm that calculated dictionary paths. We also tried variations that involved including the dictionary definitions of homonyms to try to improve our accuracy, but it didn't change our results very much.

We also used semantic similarity methods on the brown corpus, using an implementation by the same author as Lesk's algorithm. We tried all three methods: path similarity, Wu-Palmer, and Leacock-Chordorow's algorithm to see if they would yield high accuracy rates. However, each method was only accurate for certain words, but none of them could consistently identify the correct word sense.

We also tried Naive Bayes approach using two bigram models similar to the method used for our part of speech models, with one bigram model for bigrams including the word before the target word and the target word, and another bigram model for the target word and the word after the target word. This too had a low accuracy rate so we did not include it in our pipeline.

4.3 Supervised LSA

Our approach uses simple LSA with a supervised learning component.

The steps are as follows:

1. Build term-document matrix
   (a) terms are words in Brown Corpus and in training data
   (b) documents are categories of Brown, and concatenated paragraphs of training data with same word sense for ambiguous word
2. Apply SVD to get U, D, V^T [4].

   Then we have the supervised aspect of the LSA.

3. For each (paragraph, ambiguous word, word sense) tuple in the training data:
    (a) create vector in the term-document space
    (b) fold this vector into the reduced semantic space of V
    (c) use cosine similarity to come up with the eigentopic most associated with the paragraph.

    Then, we have built an association between each word sense and an eigentopic vector in V.

4. For each (paragraph, ambiguous word) in testing data:
    (a) come up with the eigentopic vector associated
    (b) For that eigentopic vector, there is a probability for each word sense for a given word – we pick the highest probability word sense for the word we are disambiguating

In other words, for a given eigentopic, we will always pick the same word sense for a given word. This approach has its flaws.

We also tried a slightly different approach to capture more of the information: For each word, and for each sense of the word, we created a vector by applying cosine similarity to each eigentopic column vector for our training vector (created from paragraph). We averaged over all the training data for each word sense. Then for testing, we created this vector from the test paragraph and compared via cosine similarity again to determine which word sense was most likely. This approach had issues and we got very bad results. It might be better in the future to try a modified approach to this (i.e. do something other than average the vectors).

5 Results

For comparison, the highest accuracy obtained in the 2001 Senseval-2 competition was 64% using a supervised approach. The best unsupervised approach had an accuracy of 40%.

5.1 POS Bigram Model

POS Bigram Model (w/frequency count > 3):

Threshold	80%	40%	0%
Correctly identified	199	518	574
Total identified	253	893	1136
% Accuracy	78.7%	58.0%	50.5%
% Coverage	5.8%	20.6%	26.2%

Threshold	80%	40%	0%
Correctly identified	466	808	850
Total identified	898	1641	1826
% Accuracy	51.9%	49.2%	46.5%
% Coverage	20.7%	38.7%	42.2%

POS Bigram Model (w/frequency count > 0)

Here are some examples of phrases that our POS Bigram model correctly disambiguates: "Marketing too, in its strictest sense, is outside our remit. " (the way in which a word or expression or situation can be interpreted)

"Before he lost his money he had the good sense to commission John Soane to design the Campden Hill Square house." (a general conscious awareness)

5.2 Supervised LSA

We give two results: with only the Brown Corpus categories as column vectors (15 column vectors $\rightarrow$ 9 column vectors after SVD) in the term-document matrix, the accuracy of word sense disambiguation was 42% (average over all the words).

With the Brown Corpus categories as well as the word-sense categories (in total, 861 column vectors $\rightarrow$ 140 column vectors after SVD), the accuracy was on average 31%, though individual accuracies for different words ranged from 7% to 61%.

6 Conclusions

6.1 Part-of-Speech Tagging

Part-of-speech performed typically very well on the subsets of the data that it deemed itself accurate enough to deal with. The main issue with POS was the subset of the data on which it was actually accurate enough to take care of. If the threshold was set at 80%, POS was applied to about only 6%of the data. Even when the threshold was set at 0%, POS only took care of 42.2% of the data, since a large number of the bigram pairs (of part-of-speech relations) never showed up in the dataset. This indicates an issue with sparsity of the data.

6.2 Supervised LSA

One issue with the LSA approach may be that the corpuses we use (namely, for a given word and a given sense, all the training paragraphs concatenated together) may actually cover a very broad array of topics, and thus a lot of noise might be added. While this should theoretically be removed by SVD, it might just be that the 'paragraph topics' were too disjoint and the correlation for different very fine-tuned usage is not strong enough.

Here are some examples of where our approach fails with Supervised-LSA:

For the word "fine":
Our program: "elegant"
Actual: "thin (in thickness)"
Our program: "superlative"
Actual: "elegant"

There are a large amount of errors in cases where it would be difficult for a human to disambiguate between the senses because the senses are very finely separated in meaning. Thus our approach falls short of disambiguating subtle nuances between word senses.

Because the LSA approach is primarily topic categorization, we need to use better documents that are more tailored to WSD.

Another issue is the sparsity of our labeled training data: because we potentially do not have enough data for a given word sense, it becomes a lot more difficult to build a valid topic model. Thus it may be that we need more outside training data that has a single ambiguous word used in a specific word sense in multiple different ways. Another necessary improvement would be the use of this word sense in a constant topic area – in other words, it would be nice if we had labeled data that used the word "fine" meaning "thin" only in sentences relating to cutting in a fine fashion (as opposed to being intermingled with sentences regarding a "fine figure", where "fine" still means "thin".)

Again, since the accuracy was highly dependent on training data, it was easier to disambiguate ambiguous words with fewer senses because there were fewer possible ways to categorize the word senses. Also, the length of the document column texts were smaller as a result for the associated word senses.

Due to the difficulty of gathering enough appropriate data, it is hard to use a topic-categorization approach to do fine-tuned WSD.

7 Future Work

There are many directions for future work with this project. Regarding just the LSA-portion of the pipeline, several improvements could be made. We could try different document columns for Supervised LSA– the method's success is in large part due to choosing document-columns appropriately. Perhaps it would make sense to create our own corpus and analyze a specific word to get really good accuracy with just disambiguating that word. We could also try improving the statistical rigor of the procedure itself by utilizing a supervised approach to PLSA or NNMF-LSA, as described earlier in the paper. We could also try an SVM approach with the supervised portion of the word – build an SVM out of the cosine vectors, and train an SVM to recognize which word sense a cosine vector is most likely to be given a cosine vector for a test data input.

The other approach we can take to extend this paper is applications. Word-Sense Disambiguation would be relevant to the field of sentiment analysis. Socher et al (2013) [8]created a sentiment analysis pipeline with Recursive Neural Tensor Networks (RNTNs). However, there is no word-sense disambiguation step, which may affect the accuracy of the process, and words like "mean" and "like" are not disambiguated correctly. We could potentially improve the accuracy of sentiment analysis by including a word-sense disambiguation step (perhaps improved from ours) into the pipeline.

8 Citations

.

1. ACL-SIGLEX. 2001. Evaluation Exercises for the Semantic Analysis of Text. Retrieved from http://www.senseval.org/.
2. Agirre, Eneko, and Philip Glenny. Edmonds. Word Sense Disambiguation: Algorithms and Applications. Dordrecht: Springer, 2006. Print.
3. Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. OReilly Media Inc.
4. Kirk Baker. 2013. Singular Value Decomposition Tutorial. Retrieved from http://www.ling.ohio-state. edu/ kbaker/pubs/Singular_Value_Decomposition_Tutorial.pdf.
5. Liling Tan. 2014. Pywsd: Python Implementations of Word Sense Disambiguation (WSD) Technologies [software]. Retrieved from https://github.com/alvations/pywsd.
6. Phil Katz and Paul Goldsmith-Pinkham. 2006. Word Sense Disambiguation Using Latent Semantic Analysis Retrieved from http://www.sccs.swarthmore.edu/users/07 /pkatz1/cs65f06-final.pdf.
7. Princeton University "About WordNet." WordNet. Princeton University. 2010. http://wordnet.princeton.edu.
8. Richard Socher, Alex Perelygin, et al. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. Retrieved from http://nlp.stanford.edu/ socherr/EMNLP2013_RNTN.pdf.
9. R. Navigli. Word Sense Disambiguation: a Survey. ACM Computing Surveys, 41(2), ACM Press, 2009, pp. 1-69. [A complete State of the Art in Word Sense Disambiguation]
10. Thomas Hofmann. 1999. Probabilistic Latent Semantic Analysis. Retrieved from http://cs.brown.edu/ th/papers/Hofmann-UAI99.pdf.
11. Tim Van de Cruys and Marianna Apidianaki. 2011. Latent Semantic Word Sense Induction and Disambiguation. Retrieved from http://aclweb.org/anthology//P/P11/P11-1148.pdf.
12. T.K. Landauer, Foltz P.W., and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, $25 : 259 - 284$.

9 Code

The code we used in this paper is available at https://github.com/alee101/wsd.