Word Sense Disambiguation using Latent Semantic Analysis

Phil Katz
Department of Computer Science
Swarthmore College
Swarthmore, PA
katz@cs.swarthmore.edu

Paul Goldsmith-Pinkham
Department of Computer Science
Swarthmore College
Swarthmore, PA
pgoldsm1@swarthmore.edu

Abstract

In this paper, we implement Latent Semantic Analysis (LSA) and apply it to the problem of Word Sense Disambiguation (WSD). We use training and test data from SENSEVAL-3 to create term-document matrices, to which we apply LSA. We combine this LSA-based classifier with a pre-existing WSD classifier combination system in an attempt to improve the accuracy of the system. Our LSA-based classifier is less effective than a comparable TF-IDF based classifier, but still effective enough to add information to the combiner.

1 Introduction

One of the fundamental tasks in natural language processing is Word Sense Disambiguation. WSD can be summarized as follows: given an ambiguous word, such as *bank*, determine which sense of the word (i.e. a financial institution, the side of a river, etc.) is being used. The most direct way to solve this problem is to use the surrounding context of an ambiguous word in a probabilistic calculation. For our disambiguation module, we extended a system that combines three context-based classifiers: a Naïve Bayesian classifier, a Decision List classifier, and a Nearest-Neighbor Cosine classifier that uses a technique called TF-IDF to increase the effect of meaningful words(Wicentowski et al., 2004).

For this paper, we are applying the technique of Latent Semantic Analysis (Landauer et al., 1998) to the problem of WSD. LSA is an approach that uses a mathematical technique called Singular Value Decomposition (SVD) that can reduce any given matrix into a set of three matrices. The rank of these matrices can then be reduced in order to maximize the amount of semantic meaning kept while decreasing the number of term dimensions. This creates a "meaning"-document matrix, which represents the dimensions of the decomposed term-document matrix that contain the most semantic value. The columns of this matrix can then be used to compare for similarity between documents. We implemented an LSA-based classifier to combine with the three classifiers already established in our system.

In Section 2, we discuss literature relevant to LSA and WSD. In Section 3, we present in detail the techniques we used to implement the LSA-based classifier and combine it with the other classifiers. In Section 4 we present the results of the LSA-based classifier. We discuss these results in Section 5.

2 Related Work

2.1 Latent Semantic Analysis

In (Landauer et al., 1998), Latent Semantic Analysis is used to reduce the dimensionality of term-document information in a similar fashion to our LSA-based classifier. However, (Landauer et al., 1998) use the reduced-dimension matrices to reconstruct the original term-document matrix, this time in a best-fit form. This information is used to compare similarities between documents and terms that were not apparent in the sparser original matrix. The goal of this operation is to determine relationships

between a set amount of document vectors; it would not be useful for determining the sense of a test document.

In our method, we instead *fold* a new document vector into the *semantic space*, and then compare this vector to the already decomposed test matrix. The purpose behind both processes are similar, since both attempt to reduce the dimensionality of the data to increase the efficiency of the data, however the implementations are different. Additionally, the focus of (Landauer et al., 1998) is on comparing the semantic space of LSA to human understanding of semantics, while our goal is much less ambitious. Given the variation of our success rates of LSA across different languages (and even different words), our results do not add much weight to the hypothesis that human semantic understanding is anything like LSA.

2.2 Word Sense Disambiguation

In (Wicentowski et al., 2004), Word Sense Disambiguation is attempted using a system of three combined classifiers, as previously enumerated. The classifications returned are fed into a classifier combiner that chooses one sense for each ambiguous word. The classifiers in our combination system were implemented to mimic those described in (Wicentowski et al., 2004).

2.2.1 Naïve Bayes

The Naïve Bayes classifier uses a basic probability equation to calculate the similarity between an instance, I, and a sense (from the training data), S_j:

$$Sim(I, S_j) = P(S_j) * P(I|S_j)$$

The classifier chooses the sense with the highest similarity score.

2.2.2 Cosine-based Clustering

The Cosine classifier creates a representative word frequency vector for each sense in the training data. It then creates a word frequency vector for each test word, and performs TF-IDF on all vectors. TF-IDF is a statistical measure that attempts to reflect the significance of a word in each document of a set, as follows:

$$\frac{w_i}{\sum_k w_k} \log \frac{|D|}{|(d_i \supset t_i)|}$$

where *D* is all documents, and $d_i \supset t_i$ is the number of documents in which term t_i appears. After applying TF-IDF to each vector, the classifier compares the angle between the word and each training sense (using cosine). The classifier chooses the training sense that is the closest to the test vector.

2.2.3 Decision List

The Decision List classifier uses the training set to determine the likelihood of correspondence between context and senses. The classifier then sorts the context words from most to least indicative and chooses the indicated sense of the most indicative word that appears in the context of a training word. If no previously seen words appear, the classifier chooses the most frequent sense. This classifier is highly dependendent on which features (part-of-speech, bigrams, etc.) are considered.

2.2.4 Classifier Combination

The method of combination used in (Wicentowski et al., 2004) is a simple voting system, where each classifier makes its best guess and the majority wins. We hope to implement a more complex combination system that relies on confidences, but the actual implementation of that system is not within the scope of this project.

3 Methods

3.1 Training the Classifiers

The classifier presented here uses training data taken from SENSEVAL-3 in Basque, Italian, Romanian, and Spanish. There are difference between the sets; for example, there are 46 ambiguous words in the Spanish set and only 39 in the Basque set. However, to prevent overfitting, we ignore the variations in the exact details of each set of data, and assume that each set has any number of ambiguous words, each with any number of senses, each with any number of training and test examples. The examples consist of part-of-speech tagged, lemmatized words for a paragraph or two surrounding the ambiguous word. We convert this data into a standard term-document matrix. Information from these vectors is used for classification by cosine similiarity, for Naïve Bayes classification, for decision list classification, and is also used to perform LSA.

3.2 Latent Semantic Analysis

In the process of Latent Semantic Analysis, Singular Value Decomposition is used to reduce a term-document matrix of occurance frequency, W, into three matrices: a left matrix, U, a diagonal matrix of eignenvalues, S, and a right matrix, V^T. [1] The matrix W is constructed with terms as the row dimension, and documents as the column dimension, representing the count of each term in each particular document. Once SVD is implemented, an $n \times m$ W matrix will be broken down into an $n \times m$ U matrix, an $m \times m$ S matrix, and an $m \times m$ V^T matrix.

Once the decomposition process is completed, the matrices are organized by magnitude of highest eigenvalues. Then, we drop low magnitude dimensions of the decomposed matrices. Choosing how many dimensions to drop was an empirical decision based off which dimension magnitude maximized the success rate for a particular language. The remaining number of dimensions is our *P-value* and selection process for P is described in the results section.

After the dimensions of the matrices have been reduced, we are left with three matrices, each with different semantic meaning. The right matrix, V^T, can be viewed as a "meaning"-document matrix, in which each document has numbers representing counts for the amount of semantic meaning within. This set of vectors can be used to disambiguate any future test document.

3.3 Applying The LSA-based Classifier

To disambiguate, we fold the targeted document into the correct semantic space so that a vector of the disambiguation target is transformed into a vector in the semantic space. This process is done by multiplying the test vector by U and S^{-1}. Once this vector is transformed, it is compared with the "meaning"-document matrix using cosine similarity to determine the most similar document. By reducing the number of term dimensions to a number of "meaning" dimensions, we are attempting to remove the noise from the term-document matrix such that the important factors in disambiguation will stand out and improve our cosine similarity scores. For clarity, a short example is provided.

3.3.1 LSA Example

Let

$$W = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 0 & 0 \\ 0 & 2 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

The columns of W each represent a document and the rows each represent a term. The individual entries represent the occurance count of each term in each document.Then, through SVD, we obtain:

$$U = \begin{pmatrix} .3849 & .265 & -.5127 \\ .3849 & .725 & .375 \\ .57735 & -.628 & .325 \\ .57735 & 0.0 & 0.0 \\ .19245 & -.097 & -.7 \end{pmatrix}$$

$$S = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2.175 & 0 \\ 0 & 0 & 1.126 \end{pmatrix}$$

$$V^T = \begin{pmatrix} .57735 & .57735 & .57735 \\ .788675 & -.57735 & -.211325 \\ .211325 & .57735 & -.788675 \end{pmatrix}$$

Now we reduce the dimension of the matrices to P=2, and obtain

$$V^{T*} = \begin{pmatrix} .57735 & .57735 & .57735 \\ .788075 & -.57735 & -.211325 \end{pmatrix}.$$

We also obtain U^* and S^*, not shown here for brevity's sake. The V^{T*} is the "meaning"-document matrix against which we compare our test vectors, where each column is a document with a particular sense attached to it. The test vectors will be compared after being folded into the semantic space by multiplying by U^* and S^{-1*}.

As for the actual data, we used data from SENSEVAL-3 for both the training and test sets. The training data contains documents that each correspond to a certain semantic sense. This data is then parsed to form a large term-document matrix which is then decomposed into the U, S and V^T matrices. The test data is parsed into individual vectors for

[1] We utilized the SVDLIBC library. This package uses the Single-Vector Lanczos method and quickly decomposed our large matrices.

each document, containing the counts for each term in the document. Within this document, there exists a single ambiguous test word to be disambiguated.

To compare the vectors, we use cosine similarity scores between each vector to determine the closeness of similarity. We started with a Nearest-Neighbor Cosine comparison classifer, but given the large number of documents to compare against, we found it more effective to implement a K-Nearest-Neighbor Cosine classifier to decide which sense corresponded best to the test vector. K-Nearest-Neighbor cosine comparison works similarily to Nearest-Neighbor, but takes the top K cosine similarity scores and sums them across each sense to determine the best sense match for each test vector. The number of neighbors, K, was determined empirically via a series of tests attempting to maximize the success rate of the LSA-based classifier.

This method of attempting to amplify the meaningful values in a vector and then compare the cosine with training vectors is similar to the TF-IDF Cosine classifier discussed in (Wicentowski et al., 2004). Because of this, we hope that our LSA-based classifier will do comparably well, and if LSA is a good way of representing the semantic meaning of a vector, our classifier should do well.

3.4 Combining the Classifiers

The ultimate goal of our disambiguation system is to combine the classifications done by a number of diverse classifiers into one guess. Because we do not yet have a good combiner, we track the information added by the LSA-based classifier by considering an *Oracle* combiner. The idea behind an Oracle is that it is the ideal combiner: if any classifier makes the correct disambiguation, the Oracle selects that classifier to "listen to" and therefore makes the correct disambiguation. Obviously, an Oracle combiner is impossible to implement, but (assuming accurate confidences), a significant improvement to the Oracle score should represent a significant amount of new information that a good combiner can use.

4 Results

The results presented in Table 3 are the precision results for the LSA-based classifer. Recall results are disregarded as meaningless, since for every test vector, we require that the classifier guess the best match possible. The LSA-based classifier was run on four seperate languages: Italian, Romanian, Basque and Spanish. For each language, the classifier gave significantly different results. In Table 4, the P-value of the classifier is varied to determine how significantly the size of the semantic spaces affects the precision. For each language, the value of K (the number of nearest neighbors to look at) was randomly chosen and held constant throughout the variation of P.

Language	LSA	LSA with TF-IDF
Italian	20.58%	**20.91%**
Romanian	**58.37%**	57.95%
Basque	**58.04%**	54.21%
Spanish	**67.46%**	67.10%

Table 1: The results of the LSA-based classifier, both with and without using TF-IDF on the original term-document matrix.

Language	Original Oracle	Oracle with LSA
Italian	64.53%	**71.14%**
Romanian	81.78%	**83.68%**
Basque	75.62%	**79.21%**
Spanish	89.13%	**90.37%**

Table 2: The improvement of the Oracle classifier using LSA.

In Table 4, precision values are additionally reported, this time using a constant P-value for each language while varying the K-value to determine the effect of changes in the number of neighbors used in the K-Nearest-Neighbor Cosine Similarity classifier.

Table 5 reports the overall best precision results for the LSA-based classifier, and compares them to the other classification systems run by our system. TF-IDF Cosine is unmistakeably the best classifier, winning in every language except Italian, where it was a close second. In comparison, the LSA-based classifer does not beat any system except for the Naïve Bayes, and only in Basque and Spanish. However, since the results are, at worst, comparable to the Naïve Bayes classifier, the LSA-based classifier should be able to add information for classifier combination.

By using TF-IDF on the original term-document

P Value	Italian (K=60)	Romanian (K=30)	Basque (K=40)	Spanish (K=40)
5	22.30%	58.26%	53.75%	**67.46%**
10	**22.58%**	57.92%	54.92%	66.84%
20	21.07%	57.81%	53.92%	66.77%
30	21.28%	**58.63%**	**55.04%**	66.34%
40	21.40%	56.93%	53.92%	66.72%
50	20.87%	57.75%	53.92%	67.27%
60	21.53%	56.62%	53.92%	66.36%

Table 3: The results of the LSA-based classifier, varying the dimensionality of the "semantic spaces" (P), for multiple languages.

K Value	Italian (P=5)	Romanian (P=20)	Basque (P=30)	Spanish (P=5)
5	17.75%	53.32%	49.08%	64.61%
10	18.00%	55.63%	52.33%	65.53%
20	19.31%	57.30%	53.83%	66.79%
30	21.03%	57.64%	53.96%	67.25%
40	21.24%	57.61%	**55.04%**	**67.46%**
50	22.10%	57.72%	54.96%	67.41%
60	**22.30%**	**58.03%**	55.00%	67.22%

Table 4: The results of the LSA-based classifier, varying the size of the K, the number of neighbors used in the cosine similarity test.

Language	Italian	Romanian	Basque	Spanish
TF-IDF Cosine	46.00%	**73.71%**	**67.58%**	**84.65%**
LSA Cosine	20.58%	58.63%	55.04%	67.46%
Decision List	**46.58%**	68.60%	58.33%	80.79%
Naïve Bayes	34.85%	62.92%	49.25%	64.62%

Table 5: Overall precision scores for each classifier by language, maximizing parameters for the LSA-based classifier.

matrix, we attempted to add extra weight to the rare and infrequent words within the matrix. However, from the results in Table 1, it is clear that using TF-IDF before the decomposition results in similar, and generally slightly worse values. There was only minimal improvement in Italian.

Fortunately, the LSA-based classifier does a good job of improving the Oracle combiner's precision. Table 2 demonstrates the improvement that the LSA-based classifier has on the Oracle combiner, showing the precision results for Oracle before and after LSA is implemented. In every language, the Oracle precision results have at least a 1% increase (Spanish) or a rougly 7% increase (Italian).

5 Discussion

Despite high hopes for the applicability of LSA to the task of sense disambiguation, the results of the LSA-based classifier were mediocre at best. Almost across the board, the classifier did at best slightly better, and usually worse, than the Naïve Bayes classifier, and consistently did significantly worse than TF-IDF cosine comparison.

5.1 Dimensional Reduction

It is clear from our results that the dimension-reduction aspect of the LSA-based classifier is less effective than implied in (Landauer et al., 1998). While it seems possible that LSA removes the noise from the original term-document matrix, it also seems to remove important disambiguation information. The fact that LSA did as well or better in some languages (Spanish and Italian) at low values of P implies that a significant reduction in the number of dimensions is necessary to obtain the best results. This, incidentally, also implies that our LSA implementation does not lack the necessary number of dimensions to accurately represent the semantic space.

5.2 Maximizing Parameters

In each language, the precision results were better with a different P-value, implying that there is no best P-value across language. Even within each individual language, the precision-maximizing P-values for individual words vary wildly. It seems unlikely that there is a universal P-value that is best for all cases; observationally, there was no apparent relation between the number of senses of an ambiguous word and the ideal P-value. The fact there there is no universal P-value for a language does not mean that LSA could not work, it simply mean that it would be difficult to get fully maximized precision out of an LSA-based classifier.

5.3 K-Nearest Neighbor counts

Unsurprisingly, the LSA-based classifier performed better with a higher number of neighbors used in the cosine similarity calculations. The fact that a value of $K = 60$ was the maximum-precision value for two languages means that a large window of cosine values improves the decision process, with more information leading to higher accuracy. We found that beyond $K = 60$, the improvement slowed to insignificant amounts.

5.4 LSA vs. TF-IDF

As discussed previously, the LSA-based classifier is very similar to the TF-IDF-based classifier. However, the different statistical reduction techniques are not the only differences between the two classifiers. The TF-IDF-based classifier compares each test vector to a single test vector created from the entire training set of a sense, while the LSA-based classifier uses K-Nearest-Neighbors. We implemented an LSA-based classifier in the style of the TF-IDF-based classifier, but the relatively low number of senses limited the dimensionality too sharply, as the results were significantly worse than chance. In future work, it would be of interest to apply a K-Nearest-Neighbors algorithm to single-document vectors on which we had performed TF-IDF.

5.5 LSA with TF-IDF

Since TF-IDF seems to be the most effective method of amplifying the semantic sense of a document, we decided to apply TF-IDF to the term-document matrix before applying LSA and see what happened. As shown in Table 1, this was not a particularly effective experiment; the results did not change significantly, but they did get slightly worse. This is not a suprising result, as TF-IDF removes much of the semantic content that LSA uses for dimensionality reduction, but it is worth noting.

5.6 Classifier Combination

Although the hope was that an LSA-based classifier would be an effective classifier on its own, we can still use it as a piece of a larger classifier combination system. Although we do not currently have such a system, we can get a sense of what the LSA-based classifier adds to such a system by looking at Table 2. Unfortunately, the only language in which the LSA-based classifier adds significant information to the Oracle combiner is Italian, the language in which the LSA-based classifier is approximately 20% accurate. This does not mean that the LSA-based classifier is completely useless, but it would require a very effective classifier combination system to utilize the knowledge gained from LSA.

6 Conclusion

Given the poor results from our LSA-based classifier, supporting the (Landauer et al., 1998) hypothesis of LSA as the form of human understanding is difficult. Rather than remove noise, the dimensionality reduction loses important disambiguation information, and prevents successful disambiguation. One potential reason is that LSA averages across documents and terms, and so rare words that could potentially aid in the disambiguation process get lost in the averaging process. It appears that TF-IDF performs so much better specifcally because it picks up on the rare words that improve the disambiguation process. In a practical sense, the conclusion from our results is very straightfoward: the more unique contextual information surrounding an ambiguous word, the easier the task of disambiguating it.

7 Future Work

There is signficant work that can still be done with the classifier system and the classifier combiner. Most of it does not fall within the scope of adding a LSA-based classifier to the system, but the methods used in building an LSA-based classifier could be applied to other classifiers in interesting ways.

- One thing that could be done is to implement a TD-IDF Cosine classifier, using single-document vectors and K-Nearest-Neighbor, as discussed previously. This could potentially improve the TF-IDF Cosine results, or, if it significantly lowered the results, it might explain why our LSA-based classifier performs so poorly in comparison with our TF-IDF-based classifier.

- Another extension to this project could be to experiment with less than 100% recall. Perhaps by allowing a classifier to only classify ambiguous words that received a score over a certain threshold, we could improve the results of our classifiers. A good combination system should include something that accomplishes this goal.

- For the LSA-based classifier, we did not experiment with using different features. We used the standard set of features that was found to maximize the results of the other classifiers: previous and next part-of-speech, and surrounding bigrams and trigrams. It is possible that removing these features and/or including others could improve the results of our LSA-based classifier.

References

T.K. Landauer, Foltz P.W, and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, 25:259–284.

Richard Wicentowski, Emily Thomforde, and Adrian Packel. 2004. The swarthmore college `senseval-3` system. In *Proceedings of Senseval-3, Third International Workshop on Evaluating Word Sense Disambiguation Systems*.