Supervised Word Sense Disambiguation using Python

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

Abstract

In this paper, we discuss the problem of Word Sense Disambiguation (WSD) and one approach to solving the lexical sample problem. We use training and test data from SENSEVAL-3 and implement methods based on Naïve Bayes calculations, cosine comparison of word-frequency vectors, decision lists, and Latent Semantic Analysis. We also implement a simple classifier combination system that combines these classifiers into one WSD module. We then prove the effectiveness of our WSD module by participating in the Multilingual Chinese-English Lexical Sample Task from SemEval-2007.

1 Introduction

One of the fundamental tasks in natural language processing is Word Sense Disambiguation (WSD). 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, a type of basketball shot, etc.) is being used. There are a number of ways to approach this problem: one simple way is to determine which sense occurs most commonly (the Most Frequent Sense, or MFS), and always guess that sense. MFS is often accepted as a baseline for lexical sample tasks. There is not a well-defined upper bound on WSD performance: (Gale et al., 1992) argue that 95% should be regarded as an absolute upper bound because even human judges do not have 100% agreement on WSD tasks for a given language and sense inventory. The approaches to WSD discussed here use the *context*, or surrounding words and syntactic features, of the ambiguous word to try to determine the correct sense.

In this thesis, we will examine a word sense disambiguation system that implements five different context-based classifiers: a Naïve Bayes classifier, a decision list classifier, a nearest neighbor cosine classifier, a k-Nearest-Neighbor cosine classifier, and a classifier based on Latent Semantic Analysis. Our system also includes a meta-classifier that combines the outputs of the stand-alone systems into one classification.

The training and test data used for the WSD system comes from SENSEVAL-3 and SemEval-2007. The system can be applied to any traditional lexical sample task, specified as follows: the training data consists of sets of sense-tagged ambiguous words, surrounded by words from the context of the ambiguous word. When applicable, the words are tagged with part-of-speech and case information. The test data is in the same format as the training data, only without the sense tags. We apply our WSD system to the following six SENSEVAL-3 lexical sample tasks: Catalan, Basque, Spanish, Italian, English, and Romanian. We also apply our system to the Multilingual Chinese-English Lexical Sample Task from SemEval-2007.

In section 2 we present in detail the implementation of each classifier and of the classifier combination system. In section 3 we present the results of each of the systems evaluated on the test data from SENSEVAL-3 and SemEval-2007. We discuss these results in section 4, followed by an examination of related literature and its contribution to our WSD system in section 5. We discuss potential areas of future work in section 6, and conclude in section 7.

2 Methods

2.1 A Lexical Sample Data Set

The six lexical sample data sets from SENSEVAL-3, and the Chinese-English set from SemEval-2007, consist of sets of training data paired with sets of test data. Each set of data consists of about 40 ambiguous words. For each ambiguous word, there are between 20 and 50 example contexts, usually including about 200 surrounding part-of-speech tagged, lemmatized words. These contexts are tagged by the sense of the ambiguous word.

The data is available in a number of different languages; here we present a discussion of systems built for the Catalan, Basque, Italian, Romanian, English, Spanish, and Chinese-English lexical sample tasks. Although there are differences between the sets (for example, there are 46 different ambiguous words in the Spanish set and only 39 in the Basque set), the classifiers are identical for all seven languages. The only practical difference is that some languages contain features that others do not (for example, Romanian is tagged with case, but English is not). These differences and the impact of using features will be discussed in Section 2.2.

The senses and corpora came from a variety of sources. In Basque, the senses came from Basque WordNet[1], and the corpus from newspapers and the internet (Agirre et al., 2004). In Italian, the senses came from the Italian MultiWordNet (Piranta et al., 2002), and the corpus from the *macro-balanced* section of the Meaning Italian Corpus (Bentivogli et al., 2003). In Spanish, the senses came from MiniDir-2.1[2], and the corpus from news articles (Màrquez et al., 2004a). In English, the senses for nouns and adjectives came from WordNet 1.7.1 (Miller, 1995), while the annotations for verbs came from Wordsmyth[3]. The English corpus consisted of examples extracted from the British National Corpus. The Romanian senses were extracted from a Romanian dictionary (Coteanu et al., 1975), while the corpus was built using the Open Mind Word Expert system (Chklovski and Mihalcea, 2002), adapted to Romanian[4]. The Catalan sense inventory came from MiniDir-Cat[5], and the corpus was extracted from news articles (Màrquez et al., 2004b).

This classifier system is also being used for the Multilingual Chinese-English Lexical Sample Task from SemEval-2007. Detailed analysis is presented from 10-way cross-validation results, and the final competition results are also presented.

As a pre-processing step, we convert the training data into a standard *term-document matrix*. We will describe the context features required for this matrix in section 2.2. This matrix contains a row for each

[1] `http://ixa3.si.ehu.es/wei3.html`

[2] MiniDir is a dictionary under development by the CLiC research group, `http://clic.fil.ub.es`

[3] `http://www.wordsmyth.net`

[4] Romanian Open Mind Word Expert can be accessed at `http://teach-computers.org/word-expert/romanian`

[5] `http://clic.fil.ub.es`

training instance of an ambiguous word, and a column for each feature that can occur in the context of an ambiguous word. Although the Naïve Bayes and decision list classifiers do not require that the training data be formatted in such a matrix, it makes implementation cleaner, and the matrix and vectors are required both for cosine comparison and for LSA.

2.2 Context Features

Throughout this paper, we will refer to the *context vector* of a given ambiguous word. A context vector is a way of keeping track of the words that occur surrounding an ambiguous word. Imagine a massive vector of integers, where each integer represents a given word. Each time that word occurs within a certain context window of an ambiguous word, the integer gets incremented. This is called a *bag-of-words* method of scoring, because it does not take the distance from the ambiguous word, or syntax of the sentence, into account. We can use more informative features, however. Imagine an even larger vector, that instead of just having words like financial, has features such as **Prev-Word-is–financial**. Specific features such as the preceding word or the part of speech of the following word can be much more informative than bag-of-words counts, as we will discuss in section 3.6.

2.3 The Classifiers

As mentioned previously, our system consists of five unique classifiers: a Naïve Bayes classifier, a decision list classifier, a nearest neighbor cosine classifier (NN-Cos), a k-Nearest-Neighbors cosine classifier (k-NN-Cos), and a classifier that uses Latent Semantic Analysis (LSA). The Naïve Bayes classifier, decision list classifier, and nearest neighbor cosine classifier, as well as the idea of simple classifier combination, are extensions of the systems presented in (Wicentowski et al., 2004b).

2.3.1 Naïve Bayes

The Naïve Bayes classifier is based on one of the simplest, most fundamental probabilistic rules: Bayes' Theorem.

$$Pr(A|B) = \frac{Pr(B|A) \cdot Pr(A)}{Pr(B)} \tag{1}$$

Given a term-document matrix, it is very straightforward to implement a Naïve Bayes classifier. The goal is to calculate, for a given context *B*, the probability of the target word being labeled as sense *A*:

Since Pr(*B*) is constant given a specific instance, and since we wish to choose the sense that maximizes the probability Pr(A|B), we can write:

$$\underset{a \epsilon senses(A)}{\operatorname{argmax}} \; Pr(A|B) = \underset{a \epsilon senses(A)}{\operatorname{argmax}} \; Pr(B|A) \cdot Pr(A) \tag{2}$$

It is simple to calculate the global probability of *A* in the training data:

$$Pr(A) = \frac{|Instances\ labeled\ as\ sense\ A|}{|Instances|} \tag{3}$$

We calculate Pr(B|A) as:

$$Pr(B|A) = \sum_{i=1}^{n} Pr(B_i|A) \tag{4}$$

where, assuming independence of the features, there are *n* different features B_i in context *B*. Pr(B_i|A) can be estimated from the training data as the frequency with which B_i occurs in the context of sense *A*.

The classifier returns the sense with the highest similarity to the test data, as show in Equation 2.

Additive Smoothing

Additive smoothing is a technique that is used to attempt to improve the information gained from low-frequency words (Chen and Goodman, 1998). We used additive smoothing in the Naïve Bayes classifier. To implement additive smoothing, we added a very small number (δ) to the frequency count of each feature (and divided the final product by this value times the size of the feature set to maintain accurate probabilities). This small number has almost no effect on more frequent words, but boosts the score of less common, yet potentially equally informative, words.

Feature	Confidence	Sense
Prev-Word-is–financial	99%	Financial
Next-Word-is–shot	98%	Basketball
WordBag-Has–bond	96%	Financial
WordBag-Has–water	95%	River
...	...	...
ELSE	40%	Financial (MFS)

Table 1: A small piece of an example decision list.

2.3.2 Decision List

A decision list is a classifier that can best be described as an extended if-then-else statement. For each matched condition, there is a single classification. Decision lists were proposed and evaluated for lexical sample tasks in (Yarowsky, 2000).

Table 1 shows an example decision list for the ambiguous English word *bank*; if the context of the test set does not have any of the features in the decision list, the classifier simply chooses the most frequent sense with a confidence of the probability of that sense.

The decision list classifier uses the log-likelihood of correspondence between each context feature and each sense, using additive smoothing (Yarowsky, 1994). The decision list was created by ordering the correspondences from strongest to weakest. Instances that did not match any rule in the decision list were assigned the most frequent sense, as calculated from the training data. The log-likelihood of correspondence is used as a confidence for classifier combination.

2.3.3 Nearest Neighbor Cosine

The nearest neighbor cosine classifier uses the context vectors created for each sense during training, and for the ambiguous instance during testing. The cosines between the ambiguous vector and each of the sense vectors are calculated, and the sense that is the "nearest" (largest cosine/smallest angle) is selected by the classifier.

2.3.4 k-Nearest-Neighbor

While the nearest neighbor cosine classifier chooses the best answer by comparing to "sense" vectors, it is also possible to choose an answer by comparing the ambiguous context vector to each individual training instance context vector. The k-Nearest-Neighbor classifier finds the *k* nearest training instances to the ambiguous test instance. The most frequent sense among the *k* nearest to the test instance is the selected sense.

TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a method for automatically adjusting the frequency of words based on their importance to a document in a corpus. TF-IDF, first introduced in (Sparck Jones, 1972), decreases the value of words that occur in multiple documents. The equation we used for TF-IDF is:

$$tf_i \cdot idf_i = n_i \cdot \log\left(\frac{|D|}{|D : t_i \epsilon D|}\right) \tag{5}$$

where n_i is the number of occurrences of a term t_i (tf), and *D* is the set of all training documents.

TF-IDF is used in the nearest neighbor and the k-Nearest-Neighbor classifiers in an attempt to minimize the noise from words such as *"and"* that are extremely common, but, since they are common across all training senses, carry little semantic content.

2.3.5 Latent Semantic Analysis

In Latent Semantic Analysis (Landauer et al., 1998), Singular Value Decomposition (SVD) is used to reduce a term-document (or term-term) matrix of term occurrences, W, into three matrices: a left matrix, U, a diagonal singular value, matrix of eigenvalues, S, and a right matrix, V. The matrix W is constructed with terms as the column dimension, and documents as the row dimension, representing the count, or some function of the count, of each term in each particular document. After applying SVD, an n x m W matrix will be decomposed into an n x m U matrix, an m x m S matrix, and an m x m V^T

matrix, so that:

$$W = U \cdot S \cdot V^T \tag{6}$$

The rows of the matrices are sorted such that the highest eigenvalues are in the top left of S. Then, all but the *n* highest values are zeroed out. Choosing the number of eigenvalues to drop is done empirically on a per-language basis using cross-validation. To do the calculations required for LSA, we used the SVDLIBC library[6]. This process is described further in section 3.5.

The reduction yields three matrices. The right matrix, V^{T*}, can be viewed as a projection of the training document set into a new m-dimensional semantic space. This matrix will be used by the LSA classifier to disambiguate any future document. By reducing the dimensionality of the semantic space, we are hoping to remove the noise from the term-document matrix so that the important factors in disambiguation will stand out and improve the accuracy of our classifier.

LSA Example

Let

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

where each column represents a document and each row represents a feature found in the documents.

[6] `http://tedlab.mit.edu/~dr/SVDLIBC/`

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 matrices dimensionally, removing all but the n highest eigenvalues (in this case, n=2), and, by multiplying the matrices back together, obtain a new matrix, V^{T*}. The columns of V^{T*} represent the documents from W folded into a new semantic space, represented by the rows of V^{T*}:

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

The V^{T*} columns are what we will use to compare new document vectors. These vectors will be folded into the semantic space using U^* and S^{-1*}.

2.3.6 The LSA Classifier

To disambiguate, the classifier will fold the target document into the reduced semantic space so that each document vector is in the same dimension as the documents in V^{T*}. This process is done by multiplying the test document vector by U and S^{-1}.

$$D^* = D^T \cdot U \cdot S^{-1} \quad (7)$$

Once this vector is transformed, it can be compared with the "meaning"-document matrix using nearest neighbor cosine similarity to determine to which document it is most similar.

2.3.7 The MFS Classifier

Used largely as a baseline, the MFS classifier simply chooses the most frequent sense from training. Short of arbitrarily picking a sense and ignoring the training data, this is the simplest baseline to compare against and is used as an "ELSE" for the decision list classifier and as a tiebreaker for the classifier combination system.

2.4 Classifier Combination

The final step of our disambiguation system is to combine the classifications done by the diverse classifiers into one answer. One way to measure the accuracy of a combiner is to compare it to an *Oracle* combiner. The idea behind an Oracle is that it is the ideal combiner: if any classifier makes the correct disambiguation, the Oracle chooses that classifier to "listen to" and therefore makes the correct disambiguation. While it may not be possible to implement an Oracle combiner, and it is not always fair to measure against an Oracle combiner (imagine a system with 100 random classifiers), an Oracle provides a reasonable upper limit here. The classifier combination algorithm used here is based on a simple voting system. Each classifier returns a score for each sense: the Naïve Bayes classifier returns a probability, the cosine-based classifiers each return a cosine, and the decision list classifier returns the weight associated with the first feature that sense has in common with the test instance. The scores from each classifier are normalized to the range [0,1], multiplied by an empirically determined constant, and summed for each sense. The combiner chooses the sense with the highest summed score. We also implemented a simple majority voting system, where the chosen sense is the sense chosen by the most classifiers, but found our weighted combination algorithm to be more accurate.

Language	MFS	NN-Cos	k-NN-Cos	LSA Cos	Dec. List	Naïve Bayes	Combined
Italian	18.30%	46.00%	40.59%	22.58%	48.50%	35.14%	**50.23**%
Romanian	58.40%	**73.71**%	63.20%	58.63%	70.04%	63.82%	72.63%
Basque	55.80%	**67.58**%	62.21%	55.04%	60.54%	52.75%	67.50%
Spanish	67.70%	**84.74**%	81.48%	67.46%	81.12%	67.01%	84.08%
Catalan	66.40%	85.40%	81.80%	66.36%	82.02%	63.78%	**85.53**%
English	55.20%	64.93%	60.24%	55.20%	62.45%	57.58%	**65.77**%
Chinese	34.99%	65.56%	61.54%	38.61%	64.37%	58.60%	**65.78**%

Table 2: Overall scores for each classifier by language. The Combined result for Chinese is the actual result of our system in SemEval-2007, and all other Chinese results are from cross-validation.

3 Results

The results presented here are measures of precision; in all cases, recall equals precision since we provide an answer for every test instance. A summary of the most important results can be found in Table 2 and Table 10.

3.1 Overall Results

Table 2 shows the overall results for our word sense disambiguation system. Except for the **Combined** result, all results listed as **Chinese** are from 10-way cross-validation on the training set from the Chinese-English Lexical Sample task, since we only have access to preliminary results from SemEval-2007 at this time. Obviously there is significant variance between languages as far as precision, but in all languages except Italian, the nearest neighbor cosine is the most precise individual classifier. The decision list classifier is also a highly effective classifier.

3.2 Naïve Bayes

The results of the Naïve Bayes classifier are improved dramatically by altering the value used in additive smoothing, as show in Table 3. The smaller δ values are generally more effective, but the changes become insignificant around 0.000001.

δ value	.01	.001	.0001	.00001	.000001
Italian	33.29%	34.81%	34.97%	**35.14%**	34.69%
Romanian	33.38%	54.54%	60.97%	62.92%	**63.82%**
Basque	24.79%	39.33%	47.33%	50.92%	**52.75%**
Spanish	54.64%	61.26%	64.00%	65.91%	**67.01%**
Catalan	54.55%	59.65%	62.14%	63.34%	**63.78%**
English	40.49%	52.18%	54.87%	56.47%	**57.58%**
Chinese	58.60%	60.80%	**61.09%**	60.95%	61.06%

Table 3: The results of the Naïve Bayes classifier across different languages with different δ values.

δ value	1	.5	.1	.05	.01	δ = 0
Italian	47.07%	47.81%	**48.50%**	47.93%	47.19%	27.59%
Romanian	69.98%	**70.04**%	69.87%	69.13%	68.71%	40.75%
Basque	**60.54**%	60.25%	59.38%	58.92%	58.54%	41.62%
Spanish	80.14%	80.81%	**81.12**%	81.00%	80.91%	56.83%
Catalan	79.89%	81.14%	81.89%	**82.02**%	81.94%	56.15%
English	62.32%	**62.45**%	61.84%	61.74%	61.51%	43.20%
Chinese	64.14%	64.37%	**64.59%**	64.48%	64.48%	44.36%

Table 4: The results of the decision list classifier across different languages with different δ values.

3.3 Decision List

As shown in Table 2, The decision list classifier performs much better than the Naïve Bayes classifier in all seven languages. Although plus-δ smoothing improves the results dramatically, the actual δ value used does not significantly affect the performance of the classifier. The decision list classifier is less appropriate for classifier combination than the other classifiers, however, because it is designed to choose only the most likely sense, rather than assign a score to each sense. This is not an insurmountable problem, as the decision list classifier does provide a score for each sense, but it is not the ideal theoretical use of a decision list.

3.4 Cosine with TF-IDF

Table 5 shows the results of the k-Nearest-Neighbor cosine classifier and compares each result to the nearest neighbor cosine classifier. Although the results of the k-Nearest-Neighbor classifier improve as *k* increases, they never approach the precision of the nearest neighbor classifier. However, since the

k-value	5	25	50	75	95	Best k-NN	NN-Cos
Italian	39.40%	40.10%	40.51%	**40.55%**	40.51%	40.55%	**46.00%**
Romanian	47.39%	57.58%	62.44%	62.61%	**63.12%**	63.12%	**73.71%**
Basque	49.46%	60.42%	**62.21%**	61.96%	62.08%	62.21%	**67.58%**
Spanish	78.55%	81.22%	**81.48**%	81.03%	80.93%	81.48%	**84.74%**
Catalan	81.31%	**81.45%**	81.27%	81.31%	81.23%	81.45%	**85.40%**
English	51.50%	58.39%	60.07%	60.12%	**60.14%**	60.14%	**64.93%**

Table 5: The results of the k-Nearest-Neighbor TF-IDF classifier across different languages with different *K* values. The last two columns show the best k-NN classifier and the classic nearest neighbor classifier.

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

nearest neighbor cosine classifier is the most precise classifier in our system, that does not mean that the k-Nearest Neighbor variant is worthless; every additional classifier has the potential to help a little bit with the combiner.[7]

3.5 LSA-based Cosine

The results presented in Table 6 and Table 7 are the precision results for the LSA classifier. This data was gathered in four of the languages: Italian, Romanian, Basque and Spanish. In Table 6, the *P*-value of the classifier (the number of eigenvalues that are not zeroed out) is varied to determine how significantly the number of "semantic spaces" affects the precision. For each language, the value of *k*, the size of the k-Nearest Neighbor algorithm, was held constant throughout the variation of *P*.

In Table 7, we use a constant *P* for each language while varying the k-value to determine the effect of

[7]Due to time constraints, we did not include a k-NN classifier on our Chinese-English system for SemEval-2007.

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

Language	LSA	LSA with TF-IDF
Italian	**22.58%**	20.91%
Romanian	**58.63%**	57.95%
Basque	**55.04%**	54.21%
Spanish	**67.46%**	67.10%

Table 8: The results of the LSA-based classifier, both with and without using TF-IDF on the original term-document matrix, for optimal values of K and P in each language.

changes in the number of neighbors used in the k-Nearest-Neighbor Cosine similarity classifier.

By using TF-IDF on the original term-document matrix, we attempted to add extra weight to the rare and infrequent words within the matrix. However, as shown in Table 8, when TF-IDF is used before the decomposition, the resulting values are consistently slightly worse.

3.6 Context Features

Unigrams, *Bigrams*, and *Trigrams* include all words in the context as unsorted n-grams. When part-of-speech tags or case tags were provided, the surrounding part-of-speech or case tag n-grams were also used with each respective feature set. *Weighting* gives added weight to the unigrams that are within a ten-word window of the ambiguous word.

Across all languages and all classifiers, our choice of features did not have a significant impact on precision. As shown in Table 9, the only classifier that was consistently impacted by feature selection was the decision list classifier. It is clear that more features results in slightly better performance, but

Italian	NN-Cos	LSA-Cos	Dec. List	Naïve Bayes	Oracle
Unigrams	**50.02%**	**23.58%**	45.55%	42.03%	75.19%
Bigrams	49.86%	22.06%	45.67%	41.98%	75.24%
Trigrams	49.98%	22.47%	45.67%	41.78%	75.52%
Weighting	45.96%	22.80%	47.03%	**43.50%**	74.91%
All	46.00%	20.83%	**48.50%**	**43.50%**	**75.40%**
Romanian	**NN-Cos**	**LSA-Cos**	**Dec. List**	**Naïve Bayes**	**Oracle**
Unigrams	72.35%	57.41%	66.73%	70.40%	82.63%
Bigrams	72.24%	**58.88%**	67.01%	70.52%	83.03%
Trigrams	72.35%	57.24%	67.01%	70.52%	82.83%
Weighting	73.60%	57.87%	69.98%	**71.08%**	**84.50%**
All	**73.71%**	58.37%	**70.04%**	71.05%	84.30%
Basque	**NN-Cos**	**LSA-Cos**	**Dec. List**	**Naïve Bayes**	**Oracle**
Unigrams	67.04%	54.58%	59.88%	58.33%	78.17%
Bigrams	67.00%	53.79%	59.88%	58.83%	78.21%
Trigrams	66.96%	54.12%	59.88%	58.79%	78.29%
Weighting	**67.58%**	54.75%	**60.54%**	62.54%	80.71%
All	**67.58%**	**55.04%**	**60.54%**	**62.62%**	**80.96%**
Spanish	**NN-Cos**	**LSA-Cos**	**Dec. List**	**Naïve Bayes**	**Oracle**
Unigrams	84.12%	66.77%	77.74%	83.15%	89.99%
Bigrams	84.29%	**67.75%**	78.19%	83.15%	90.46%
Trigrams	84.34%	67.13%	78.19%	83.12%	90.46%
Weighting	**84.74%**	67.03%	80.12%	83.53%	91.04%
All	**84.74%**	67.56%	**81.12%**	**83.50%**	**91.20%**
Catalan	**NN-Cos**	**LSA-Cos**	**Dec. List**	**Naïve Bayes**	**Oracle**
Unigrams	84.38%	65.78%	77.50%	84.20%	90.77%
Bigrams	84.55%	66.22%	77.76%	84.11%	90.99%
Trigrams	84.55%	65.91%	77.76%	84.11%	90.77%
Weighting	**85.49%**	**66.49%**	79.94%	84.24%	**91.88%**
All	85.40%	66.36%	**82.02%**	**84.29%**	91.92%
English	**NN-Cos**	**LSA-Cos**	**Dec. List**	**Naïve Bayes**	**Oracle**
Unigrams	**65.42%**	54.82%	62.53%	64.30%	78.78%
Bigrams	65.29%	55.17%	62.63%	64.45%	**79.21%**
Trigrams	65.39%	54.46%	62.63%	64.43%	78.52%
Weighting	64.88%	54.61%	62.32%	**64.81%**	78.96%
All	64.93%	**55.20%**	**62.45%**	64.71%	79.08%
Chinese	**NN-Cos**	**LSA-Cos**	**Dec. List**	**Naïve Bayes**	**Oracle**
Unigrams	58.82%	36.88%	57.65%	55.88%	83.24%
Bigrams	60.42%	**38.72%**	59.98%	55.55%	**85.59%**
Trigrams	60.39%	38.61%	59.98%	55.36%	85.44%
Weighting	**63.03%**	38.05%	62.77%	**58.60%**	85.15%
All	62.92%	38.16%	**62.88%**	**58.60%**	85.56%

Table 9: Impact of features on classifier performance across all languages. The features listed are not additive.

Language	Oracle	Voting Combiner	Confidence Combiner
Italian	75.40%	48.83%	**50.23%**
Romanian	84.30%	71.00%	**72.63%**
Basque	80.96%	64.54%	**67.50%**
Spanish	91.20%	81.45%	**84.08%**
Catalan	91.92%	82.20%	**85.53%**
English	79.08%	63.59%	**65.77%**
Chinese	83.82%	61.76%	**67.38%**

Table 10: Classifier Combination results for each language.

this correlation is not nearly as strong as expected.

3.7 Classifier Combination

As discussed previously, two classifier combiner systems are evaluated. The first is a simple voting system that takes the best guess of each classifier and chose the most common guess. The second was a variant on the voting system that weighted the guesses of each classifier on each sense (according to the Naïve Bayes probability, decision list confidence, or actual cosine score) and then multiplied those guesses by an empirically determined constant for each classifier. As shown in Table 10, this *confidence combiner* achieved consistently better results than the simple voting combiner.

3.8 SENSEVAL-3 Results

The results of the SENSEVAL-3 competition are publicly available, so it is possible for me to compare our disambiguation system with the systems that were entered in SENSEVAL-3. Our disambiguation system is inserted as if it were entered in the competition and labeled as **PK-Dispy**.

Table 11 shows the results from the Spanish and the Basque lexical sample tasks from SENSEVAL-3. The TF-IDF cosine classifier would have been the most precise classifier in the Spanish task, and in the top half of the Basque results. The combined system would have been the second-most precise Spanish system and the fourth-most precise Basque system.

Table 12 shows the results from the Catalan, Romanian, and Italian lexical sample tasks.

Table 13 shows the results from the English lexical sample task. It is difficult to judge the results of

Spanish	Precision	Basque	Precision
MFS	67.7%	MFS	55.8%
IRST	84.2%	basque-swat-hk-bo	71.1%
PK-Dispy	**84.1%**	BCU-Basque-svm	69.9%
UA-SRT	84.0%	BCU-Basque-Comb	69.5%
UMD	82.5%	**PK-Dispy**	**67.5%**
UNED	81.8%	swat-hk-basque	67.0%
SWAT	79.5%	IRST-Kernels-bas	65.5%
D-SLSS	74.3%	swat-basque	64.6%
CSUSMCS	67.8%	Duluth-BLSS	60.8%
UA-NSM	61.9%	UMD-SST1	65.6%

Table 11: SENSEVAL-3 results in Spanish and Basque, with our combination system results added

Catalan	Precision	Romanian	Precision	Italian	Precision
MFS	66.4%	MFS	58.4%	MFS	18.3%
IRST	85.8%	romanian-swat-hk-bo	72.7%	IRST-Kernels	53.1%
PK-Dispy	**85.5%**	**PK-Dispy**	**72.6%**	swat-hk-italian	51.5%
SWAT-AB	83.4%	swat-hk-romanian	72.4%	**PK-Dispy**	**49.9%**
UNED	81.9%	Duluth-RLSS	71.4%	UNED	49.8%
UMD	81.5%	swat-romanian	71.0%	italian-swat-hk-bo	48.3%
SWAT-CP	79.7%	UMD-SST6	70.7%	swat-italian	46.5%
SWAT-CA	79.6%	ubb-nbc-ro	71.0%	IRST-Ties	39.6%
Duluth-CLSS	75.4%	UBB	67.1%	———-	—

Table 12: SENSEVAL-3 results in Catalan, Romanian, and Italian, with our combination system results added

English	Precision	(continued)	Precision	(continued)	Precision
htsa3	72.9%	MC-WSD	71.1%	SyntaLex-3	64.6%
IRST-Kernels	72.6%	HLTC-HKUST-all2	70.9%	UNED	64.1%
nusels	72.4%	NRC-Fine	69.4%	SyntaLex-4	63.3%
htsa4	72.4%	HLTC-HKUST-me	69.3%	CLaC2	63.1%
BCU-comb	72.3%	NRC-Fine2	69.1%	SyntaLex-1	62.4%
htsa1	72.2%	GAMBL	67.4%	SyntaLex-2	61.8%
rlsc-comb	72.2%	SinequaLex	67.2%	UJAEN	61.3%
htsa2	72.1%	CLaC1	67.2%	R2D2	63.4%
BCU-english	72.0%	SinequaLex2	66.8%	**MFS**	**55.2%**
rlsc-lin	71.8%	UMS-SST4	66.0%	IRST-Ties	50.2%
HLTC-HKUST-all	71.4%	**PK-Dispy**	**65.8%**	NRC-Coarse	48.5%
TALP	71.3%	Prob1	65.1%	NRC-Coarse2	48.4%

Table 13: SENSEVAL-3 results in English, with our combination system results added

the English task because there are so many entries, and because most disambiguation systems submitted multiple entries (with tweaked parameters). Our results are far from the best, but they are slightly above average, taking all the multiple entries into account.

4 Discussion

The reason that we implemented this word sense disambiguation system was to recreate and improve upon the results discussed in (Wicentowski et al., 2004b). We implemented the three classifiers presented there, and added both the SVD classifier and the k-NN classifier, and we improved upon the classifier combination system. The results of our system compare favorably with the results from SENSEVAL-3. The results with titles including *swat-hk* are not comparable to our system; they took the results of the *swat* system and applied an algorithm called Boosting (Wicentowski et al., 2004a) that will not be discussed further in this thesis.

4.1 Naïve Bayes

Empirically, the Naïve Bayes classifier is less precise than our best classifiers. The results of the Naïve Bayes classifier are above the MFS baseline, and so the classifier is doing something more effective than blind guessing, and is therefore a worthy addition to the combination system.

4.2 Decision List

The decision list classifier is an accurate classifier. In Italian, it is the most precise classifier, and in every language except Basque it is within a few percentage points of the most precise classifier.

4.3 Cosine-based Clustering

Cosine-based clustering includes the most valuable method in our system. Our results show that the LSA-based clustering is not particularly effective, and the k-Nearest-Neighbors clustering is at best a weaker version of the classic nearest neighbor clustering. The nearest neighbor clustering is the most

precise classifier in five of the seven language tasks we attempted. In three of the languages, it is even more precise than the combined classifier; in other words, combining the NN classifier with other classifiers only weakens the overall classification.

4.3.1 LSA

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 certain languages (Spanish and Italian) at low values of P implies that it was not lacking the dimensions to represent the semantic space (see Table 6), as more dimensions did not improve the results.

4.3.2 k-Nearest-Neighbor Cosine

As mentioned previously, the k-Nearest-Neighbor classifier on individual instances is less precise than the classic nearest neighbor algorithm on sense vectors. For LSA, it is necessary to do k-NN because otherwise there are not enough dimensions to do singular value decomposition (the P value can only be as large as the number of documents in order for the matrix calculations to work), but it does not seem to be particularly productive to implement a unique k-NN when a classic nearest neighbor algorithm can be used instead.

4.4 Classifier Combination

Classifier combination is one area in which our system could be significantly improved. In Spanish and especially Catalan, our combiner does a fairly good job of utilizing the available knowledge, as measured by the gap between the combiner and an Oracle classifier. In other languages, however, the gap is more significant.

5 Related Work

5.1 Word Sense Disambiguation

In (Ide and Veronis, 1998), the authors describe the history of Word Sense Disambiguation and the approaches that have been used to disambiguate word senses. Mirroring the growth of the larger Artificial Intelligence field, the evolution of WSD systems begins at symbolic semantic networks, passes through connectionist neural network models, knowledge-based models, and arrives at the "modern" approaches to WSD: dictionary- and thesaurus-based models, and corpus-based methods. The focus of (Ide and Veronis, 1998) then shifts to WSD evaluation, discussing *in vitro* evaluation, exemplified by the SENSEVAL competitions, where the results of a WSD system are compared to a gold standard of correct answers. This is contrasted with *in vivo* evaluation, where the results of a WSD system are only measured by the amount that they contribute to a larger application, such as machine translation and information retrieval.

5.2 Bounds On WSD Performance

(Gale et al., 1992) discuss the upper and lower bounds on "the level of performance that can be expected in an evaluation". The lower bound they discuss is the generally accepted lower bound in WSD tasks: choosing the most frequent sense (MFS) of an ambiguous word. However, one significant problem with an MFS lower bound is that it requires knowledge of the distribution of senses of each ambiguous word; without a sufficiently large training set, this is not trivial knowledge to obtain. An MFS lower bound also does not take into account any value that high precision/low recall systems might have; for a system where such a tradeoff is acceptable, it is unclear that an MFS lower bound is appropriate

The authors argue that the upper bound for any WSD system should be about 95%, because even human annotators do not agree on the sense of every single word. They found an agreement rate of 96.8% over 5 human judges on 82 instances of two-sense polysemous words, and that rate seems likely to drop with more senses per word.

5.3 Unsupervised WSD

In (Yarowsky, 1995), unsupervised methods of WSD are discussed. Two fundamental rules are necessary for this system: One sense per collocation, and one sense per discourse. A collocation refers to a specific contextual trait of an ambiguous instance, such as **preceded-by-muddy**, which, for the target word *bank* would seem likely to always refer to the "river bank" sense. A discourse refers to a given document: in an essay about mutual funds, it is unlikely that river banks would be referenced. These rules are as straightforward as they appear: One sense per collocation predicts that for any given collocation involving a polysemous word, all instances have the same sense. One sense per discourse predicts that in a given discourse, all instances of a polysemous word share the same sense. One sense per discourse could fail in a case where, for example, a financial bank is located by the side of a river, but in actual text, such pathological cases should be quite rare.

Relying on those two simple rules, and a few hand-chosen seeds, (Yarowsky, 1995) uses a bootstrapping method to group the instances of a polysemous word into senses.

5.4 Hierarchical Decision Lists

In (Yarowsky, 2000), a decision list algorithm for WSD is proposed and evaluated. The algorithm uses collocational features such as the surrounding n-grams and surrounding parts-of-speech. (Yarowsky, 2000) describes a number of additional features for the decision lists, including common collocations and specific syntactic features of the ambiguous word. This algorithm is extremely effective on the SENSEVAL data set, which is why we attempted to clone it for our own system.

5.5 Using TF-IDF

In (Robertson, 2004), the term weighting function Inverse Document Frequency (IDF) is discussed and analyzed. IDF was first introduced in (Sparck Jones, 1972), and has since been proven effective in a variety of uses. In WSD, IDF is often applied as part of a IDF weighting scheme. (Robertson, 2004) delves into information theory and probability models to attempt to explain and justify the well-documented

success of IDF, ultimately concluding that IDF is "neither a pure heuristic, nor the theoretical mystery many have made it out to be." (Robertson, 2004) argues that TF-IDF is in fact grounded in probability theory and is not only effective but also justified.

5.6 Latent Semantic Analysis

In (Landauer et al., 1998), Latent Semantic Analysis is used to reduce the dimensionality of term-document information. In contrast to the system presented in this thesis, (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 may not have been apparent in the original sparse matrix. While this approach is useful to determine relationships between a fixed number of document vectors, it is less useful in a generative sense: the matrix decomposition would have to be completely recalculated each time a new document is introduced.

In our method, we instead *fold* a new document vector into the semantic space, and then compare this vector to the already decomposed matrix (as described in Section 2.3.6). The purpose behind both processes are similar, since both attempt to reduce the dimensionality of the data to remove non-semantic content, although the implementations are different. Additionally, the focus of (Landauer et al., 1998) is on comparing LSA's similarity to human understanding of semantics, while our view is much less ambitious. Given the variation of success of LSA across different languages (and even different words), our results do not add much weight to the hypothesis that human semantic understanding is like LSA.

5.7 WSD at Swarthmore

In (Wicentowski et al., 2004b), 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. Our classifiers were implemented to mimic and extend those described in (Wicentowski et al., 2004b), and to attempt to improve upon the results presented therein. The classifiers used in (Wicentowski et al., 2004b) were a Naïve Bayes classifier, a nearest neighbor cosine classifier, and a decision list classifier. The method of combination used in

(Wicentowski et al., 2004b) is a simple voting system, where each classifier makes its best guess and the majority wins.

5.8 Using Parallel Texts for WSD

In (Gale et al., 1993), the problem of a limited number of annotated sources is discussed; the authors propose a method of using pre-existing parallel texts in multiple languages as annotated sources. The example they discuss is of the word *sentence* in an English text, translated to *peine* (a judicial sentence) in some occurrences and translated to *phrase* (a syntactic sentence) in others. By using these translations as "senses" to disambiguate, the authors dramatically increase their annotated sources.

6 Future Work

There is significant work that can still be done with the classifier system and the classifier combiner.

- One way to extend this system would be to continue to experiment with changes to the classifiers; for example, the nearest neighbor cosine classifier could implement a Mutual Information algorithm instead of TF-IDF, as in (Yarowsky, 1992).

- The decision list classifier could be modified to implement aspects of decision trees in the style of (Yarowsky, 2000).

- A more "intelligent" classifier combination system could be utilized; something like a neural net or a genetic algorithm that is designed to find patterns in data could be particularly effective if it could be trained appropriately.

- Another extension to this system 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 precision of our classifiers. A good combination system should have a way to measure the confidence of each classification, so this should be an achievable goal.

- It would be interesting to apply this system to *in vivo* tasks, such as the Web People Search or Metonymy Resolution tasks from SemEval-2007.

7 Conclusion

Word Sense Disambiguation is one of the fundamental tasks in computational linguistics. Its applications include machine translation and information retrieval. In this paper, we describe and analyze five classifiers for lexical sample problems in WSD, and a classifier combination system. Our results are comparable with the state-of-the-art at SENSEVAL-3 and among the top three at SemEval-2007. Although we do not introduce any revolutionary methods, we show that good results can be obtained with a fairly straightforward combination of currently existing methods.

References

Eneko Agirre, Itziar Aldabe, Mikel Lersundi, David Martinez, Eli Pociello, and Larraitz Uria. 2004. The basque lexical-sample task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 1–4.

Luisa Bentivogli, Christian Girardi, and Emanuele Piranta. 2003. The MEANING Italian Corpus. In *Proceedings of the Corpus Linguistics Conference*, pages 103–112.

S. F. Chen and J. Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Harvard University.

T. Chklovski and R. Mihalcea. 2002. Building a sense tagged corpus with Open Mind Word Expert. In *Proceedings of the Workshop on "Word Sense Diambiguation: Recent Successes and Future Directions"*.

I. Coteanu, L. Seche, M. Seche, A. Burnei, E. Ciobanu, E. Contraş, Z. Creţa, V. Hristea, L. Mareş, E. Stîngaciu, Z. Ştefănescu, T. Ţugulea, I. Vulpescu, and T. Hristea. 1975. Dicţionarul Explicativ al Limbii Române. Editura Academiei Republicii Socialiste România.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. Estimating Upper and Lower Bounds on the Performance of Word-Sense Disambiguation Programs. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1993. A Method for Disambiguating Word Senses in a Large Corpus. *Computers and Humanities*, 26:415–439.

Nancy Ide and David Veronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, 24(1):1–40.

T.K. Landauer, Foltz P.W, and D. Laham. 1998. Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:259–284.

L. Màrquez, M. Taulé, M.A. Martí, M. García, N. Artigas, F.J. Real, and D. Ferrés. 2004a. Senseval-3: The Spanish Lexical Sample Task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 21–24.

L. Màrquez, M. Taulé, M.A. Martí, M. García, F.J. Real, and D. Ferrés. 2004b. Senseval-3: The Catalan Lexical Sample Task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 147–150.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Emanuele Piranta, Luisa Bentivogli, and Christian Girardi. 2002. MultiWordNet: developing an aligned multilingual database. In *Proceedings of the First International Conference on Global WordNet*, pages 293–302.

Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Richard Wicentowski, Grace Ngai, Dekai Wu, Marine Carpuat, Emily Thomforde, and Adrian Packel. 2004a. Joining Forces to Resolve Lexical Ambiguity: East Meets West in Barcelona. In Rada Mi-

halcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 262–264.

Richard Wicentowski, Emily Thomforde, and Adrian Packel. 2004b. The Swarthmore College SENSEVAL-3 System. In *Proceedings of Senseval-3, Third International Workshop on Evaluating Word Sense Disambiguation Systems*.

David Yarowsky. 1992. Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. In *Proceedings of COLING-1992: The 14th International Conference on Computational Linguistics*, volume 2, pages 454–460.

David Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 89–96.

David Yarowsky. 2000. Hierarchical decision lists for word sense disambiguation. *Computers and the Humanities*, 34(2):179–186.