A Multiclassifier based Approach for Word Sense Disambiguation using Singular Value Decomposition

Ana Zelaia, Olatz Arregi and Basilio Sierra
Computer Science Faculty
University of the Basque Country
ana.zelaia@ehu.es

Abstract

In this paper a multiclassifier based approach is presented for a word sense disambiguation (WSD) problem. A vector representation is used for training and testing cases and the Singular Value Decomposition (SVD) technique is applied to reduce the dimension of the representation. The approach we present consists in creating a set of k-NN classifiers and combining the predictions generated in order to give a final word sense prediction for each case to be classified. The combination is done by applying a Bayesian voting scheme. The approach has been applied to a database of 100 words made available by the lexical sample WSD subtask of SemEval-2007 (task 17) organizers. Each of the words was considered an independent classification problem. A methodological parameter tuning phase was applied in order to optimize parameter setting for each word. Results achieved are among the best and make the approach encouraging to apply to other WSD tasks.

1 Introduction

Word sense disambiguation (WSD) is the problem of determining which sense of a word is used when a word appears in a particular context. In fact, WSD is an important component in many information organization tasks, and fundamentally consists in a classification problem: given some word-contexts corresponding to some possible senses, the WSD system has to classify an occurrence of the word into one of its possible senses.

In the approach presented in this paper, a vector representation is used for training and testing word cases and the Singular Value Decomposition of matrices is applied in order to reduce the dimension of the representation. In particular, Latent Semantic Indexing (LSI) [2] is used to make the dimension reduction. This technique compresses vectors representing word related contexts into vectors of a lower-dimensional space and has shown to have the ability to extract the relations among features representing words by means of their context of use.

We present a multiclassifier [8] based approach which uses different training databases. These databases are obtained from the original training dataset by random subsampling. The implementation of this approach is made by a model inspired in bagging [3], and the k-NN classification algorithm [4] is used to make sense predictions for testing words.

For experimentation, a previous tuning phase was performed to training data in order to automatically set some system parameters to their optimal values. Four are the parameters to be optimized, and the combination of all of them gives the possibility to perform the complete disambiguation process by 1440 different ways for each of the 100 words to be disambiguated. The tuning phase has been performed in a sound manner with the aim to improve our previous work [10]. Although the computational payload is high, it is a systematic way to fix the optimal values for parameters.

The aim of this article is to give a brief description of our approach to deal with the WSD task and to show the results achieved. In Section 2, our approach is presented. In Section 3, the experimental setup is introduced. The experimental results are presented and discussed in Section 4, and finally, Section 5 contains some conclusions and future work.

2 Proposed Approach

In this section, our approach is presented and the techniques used are briefly reviewed. First the dataset used in our experiments is described and previous results are presented. Next, the data preparation is explained in more detail. A short introduction to the SVD theory and to the k-NN classification algorithm is given afterwards. Finally, the multiclassifier construction is shown.

2.1 Dataset and previous results

The dataset we use in the experiments was obtained from the 4th International Workshop on Semantic Evaluations (SemEval-2007) web page[1], task 17, subtask 1: Coarse-grained English Lexical Sample WSD. This task consists of lexical sample style training and testing data for 100 lemmas (35 nouns and 65 verbs) of different degree of polysemy (ranging from 1 to 13) and number of instances annotated (ranging from 19 instances in training for the word *grant* to 2536 instances at *share*).

The average inter-annotator agreement for these lemmas is over 90%. In [9] task organizers describe the results achieved by the participating systems. They define a baseline for the task based on giving the most frequent sense in training (F-score: 78.0%). The best system performance (89.1%) was closely approaching the inter-annotator agreement but still below it.

2.2 Data Preparation

Once we downloaded the training and testing datasets, some features were extracted and vector representations were constructed for each training and testing case. The features were extracted by [1] and are local collocations (bigrams and trigrams formed with lemmas, word-forms or PoS tags around the target), syntactic dependencies (using relations like object, subject, noun modifier, preposition and sibling) and Bag-of-words features. This way, the original training and testing databases were converted to feature databases.

2.3 The SVD technique using LSI

The SVD technique consists in factoring term-document matrix M into the product of three matrices, $M = U \Sigma V^T$ where Σ is a diagonal matrix of singular values, and U and V are orthogonal matrices of singular vectors (term and document vectors, respectively). Being k the number of singular values in matrix Σ and selecting the p highest singular values $p < k$, a vector representation for the training and testing cases can be calculated in the reduced dimensional vector space $\mathbb{R}^p$.

In our experiments we construct one feature-case matrix for each of the 100 words using the corresponding feature training dataset. Each of the columns in this matrix gives a vector representation to each of the training cases. As the number of training cases varies among different words, the number of columns present in the matrices is different; consequently, the

[1]http://nlp.cs.swarthmore.edu/semeval/tasks/task17/data.shtml

number of singular values changes as well. Taking this in consideration, we calculate the SVD of each matrix and obtain the reduced vector representations for training and testing cases for different p values. In order to calculate the SVD of the matrices, we use Latent Semantic Indexing (LSI) [2] [5], which has been successfully used for classification purposes [7],

2.4 The k-NN classification algorithm

k-NN is a distance based classification approach. According to this approach, given an arbitrary testing case, the k-NN classifier ranks its nearest neighbors among the training cases [4].

In the approach presented in this article, the training and testing cases for each word are represented by vectors in each reduced dimensional vector space. The nearest to a testing case are considered to be the vectors which have the smallest angle with respect to it, and thus the highest cosine. That is why the cosine is usually calculated to measure the similarity between vectors. The word senses associated with the k top-ranking neighbors are used to make a prediction for the testing case. Parameter k was optimized for each word during tuning phase.

2.5 The multiclassifier construction

The combination of multiple classifiers has been intensively studied with the aim of improving the accuracy of individual components [8]. A widely used technique to implement this approach is *bagging* [3], where a set of training databases TD_i is generated by selecting n training cases drawn randomly with replacement from the original training database TD of n cases. When a set of $n_1 < n$ training cases is chosen from the original training collection, the bagging is said to be applied by random subsampling.

In our work, we construct a multiclassifier by applying random subsampling for each word. As the number n of training cases is different for each word, we optimize via tuning the parameter n_1 for each multiclassifier constructed. This way, we work with training databases TD_i of different sizes. Moreover, the number of training databases TD_i to create for each multiclassifier, is also optimized via tuning.

Once the multiclassifiers are constructed, and given a testing case q for a word, the corresponding multiclassifier will make a word-sense label prediction c^i based on each one of the training databases TD_i. In order to calculate these confidence values, word-sense predictions are made for training cases

[2] http://lsi.research.telcordia.com, http://www.cs.utk.edu/∼lsi

and the accuracies obtained give the confidence values which indicate the accuracy level that may be expected when a prediction is made for a testing case based on each training database TD_i and word-sense c_j to be predicted. The way we combine such predictions is by applying Bayesian voting [6], where a confidence value $\mathrm{cv}^i_{c_j}$ is calculated for each training database TD_i and word-sense c_j to be predicted. In testing phase, confidence values obtained for the testing cases are summed by sense; the sense c_j that gets the highest value is finally proposed as a prediction for the testing case q. This process is repeated for every testing case.

In Fig. 1 an illustration of the experiment performed for each one of the 100 words can be seen. First, vectors in the original Vector Space are projected to the reduced space using SVD; next, random subsampling is applied to the training database TD to obtain different training databases TD_i; afterwards, the k-NN classifier is applied for each TD_i to make sense label predictions; finally, Bayesian voting scheme is used to combine predictions, and c will be the final sense label prediction for testing case q.

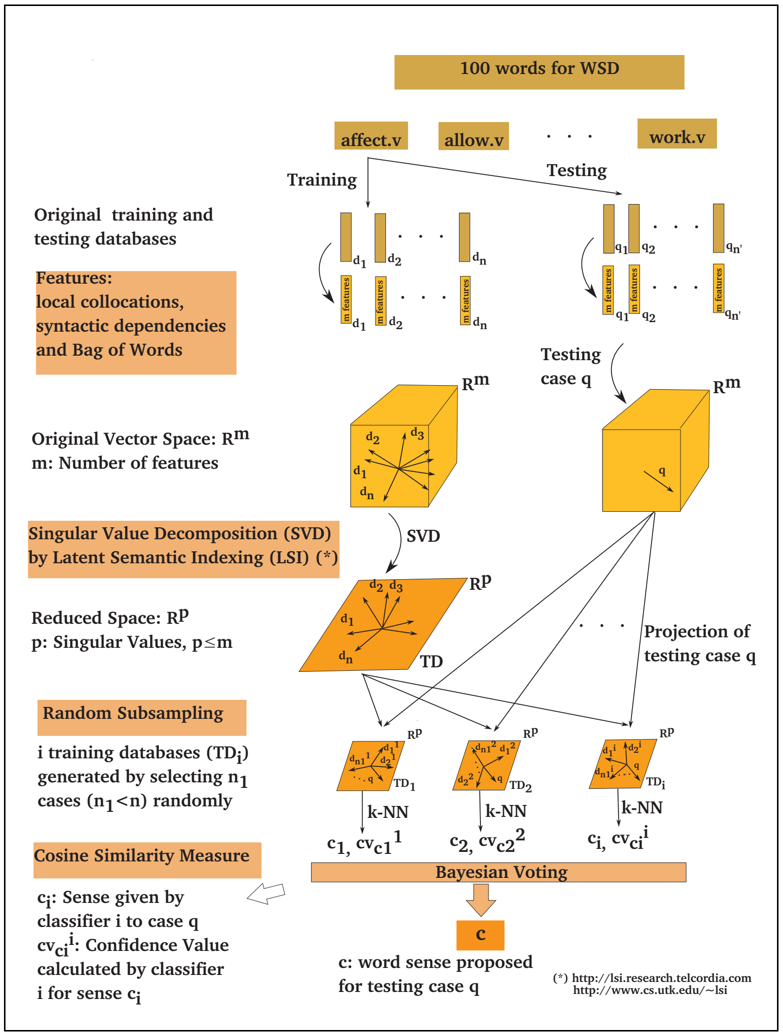
3 Experimental Setup. The tuning phase

The experiments were carried out in two phases. First, a parameter tuning phase was performed in order to set the following parameters to their optimal values:

- The dimension p of the reduced dimensional vector space $\mathbb{R}^p$ to which word-case vectors are projected for each word.
- The number of classifiers, training databases TD_i, to create for each word.
- The number k of nearest neighbors to be considered by the k-NN classifier for each word.
- The number n_1 of cases to select from the TD of each word in order to create each one of the TD_i, that is, the size of each TD_i.

All the four parameters were adjusted independently for each word, because of the different characteristics of words with respect to the number of training and testing cases present in the dataset and the number of word-senses associated to each of them.

Validation and testing data subsets used in the tuning phase were extracted form the original training database TD for each word. Both subsets



100 words for WSD
affect.v
allow.v
work.v
Training
Testing
Original training and testing databases
Features: local collocations, syntactic dependencies and Bag of Words
m features
Testing case q
R^m
Original Vector Space: R^m
m: Number of features
Singular Value Decomposition (SVD) by Latent Semantic Indexing (LSI) (*)
SVD
R^p
Reduced Space: R^p
p: Singular Values, p≤m
TD
Projection of testing case q
Random Subsampling
i training databases (TD_i) generated by selecting n_1 cases (n_1<n) randomly
k-NN
c_1, cv_c1^1
c_2, cv_c2^2
c_i, cv_ci^i
Cosine Similarity Measure
Bayesian Voting
c
c_i: Sense given by classifier i to case q
cv_ci^i: Confidence Value calculated by classifier i for sense c_i
c: word sense proposed for testing case q
(*) http://lsi.research.telcordia.com
http://www.cs.utk.edu/~lsi

Figure 1: Proposed multiclassifier approach for WSD task

were constructed by random selection of cases, where 75% of the cases were selected for the validation subset and the rest for the tuning purposed made testing subset.

In the following the optimization of parameters is explained. Parameters were optimized in the same order as presented in this subsection, that is, the dimension reduction first, the number of classifiers second, the number k of nearest neighbors third and the size of each TD_i last. When the first parameter was being optimized, all possibilities for the other three parameters were taken into account, and the optimization of the parameter was made based on the average of the 10% best results. Once a parameter was fixed, the same method was applied in order to optimize the rest of the parameters. This optimization method implies that the experiment was performed for all the combinations of the four parameters. This implies a high computational cost during the tuning phase. For testing phase, the experiments are performed using the optimal values for parameters.

3.1 The dimension p of $\mathbb{R}^p$

This is the first parameter we tuned. As it was previously mentioned in Section 2.3, the dimension p of the reduced dimensional vector space $\mathbb{R}^p$ to which training and testing cases are projected varies for different words. The reason for that is the difference in the number of cases present in the dataset for each word. For words with a high number of cases, the dimension was previously reduced to 500 (see [2]). Then, for every word we experimented by keeping the number of dimensions in a proportion. This proportion is given by parameter λ. We analyze four proportions by setting parameter λ to: $\lambda = 0$ keep all dimensions, $\lambda = 1$ keep 2/3 of the dimensions, $\lambda = 2$ keep half of the dimensions and $\lambda = 3$ keep a third of the dimensions. We calculated four different values for p. Training and testing cases were represented in the four $\mathbb{R}^p$ spaces and word-sense label predictions calculated for all of them. All the possibilities were tried for the rest of the parameters (detailed in the following subsections). For each value of λ, we selected the 10% best results from the 1440 we have, calculated the average of them and set parameter λ to its optimal value for each word. The optimization of λ gives a final optimal value for parameter p for each word.

3.2 The number of classifiers, TD_i

The number of classifiers, or TD_i to create for each word is also a parameter that needs to be tuned. This is because the number of cases present for each word is quite variable, and this fact may have some influence in the number of TD_i to construct. In our work, we experimented with 6 different values for parameter $i = 3, 5, 10, 20, 30, 40$. We performed the disambiguation process for each of them by considering the results for the optimal value of parameter λ, already optimized, and all the possible values for the rest of the parameters for each word. We then selected the best 10% average results achieved for each value of i, calculated the average, and based on these average results set the optimal value for parameter i for each word.

3.3 The number k of nearest neighbors for k-NN

At this stage of the tuning phase, and having already optimized the dimensionality reduction and the number of classifiers to create for each word, we take both optimal values and experiment with all possible values for the rest of the parameters. We calculate the average for six different values of k, $k = 3, 5, 7, 9, 11, 13$. We set the optimal value of k for each word based on the maximum average obtained.

3.4 The size of training databases TD_i: parameter n_1

As it was mentioned in Section 2.5, the parameter n_1 will be optimized for each word in order to create training databases TD_i of different sizes. The selection of different values for n_1 was experimented for each word according to the following equation:

$$n_1 = \sum_{i=1}^{s} (2 + \lfloor \frac{t_i}{j} \rfloor), \qquad j = 1, \ldots, 10$$

where t_i is the total number of training cases in the sense c_i and s is the total number of senses for the given word. By dividing t_i by j, the number of training-cases selected from each word-sense preserves the proportion of cases per sense in the original one. However, it has to be taken into account that some of the word-senses have a very low number of training-cases assigned to them. By summing 2, at least 2 training-cases will be selected from each word-sense. In order to decide the optimal value for j, the classification experiment was carried out varying j from 1 to 10 for each word. Given that parameters p, i and k are already set to their optimal values for each word, we calculate results for the 10 possible values of j, and set it to its optimal value.

4 Experimental Results

The experiment was conducted by considering the optimal values for parameters tuned. Original training and testing datasets were used for the final experiment, and results achieved were compared to the ones made available by task organizers [9].

Our system achieved an F-score of 85.65%, which compared to the baseline defined (78.0%) is a very good result, although still below the best published by task organizers (89.1%).

In [9] the performance of the top-8 systems on individual verbs and nouns is shown; 73 of the 100 lemmas are included in a table in two separated groups. Lemmas that have perfect or almost perfect accuracies have been removed. In TABLE 1 the average results achieved by our system for the two groups of lemmas are compared to the ones published in the cited paper. We can observe that our system performs better than the average of the top-8 systems disambiguating nouns, but slightly worse for verbs. In the overall, our system is very near to the average performance of the top-8 systems.

	Top-8	Our system
Verbs	70.44	67.78
Nouns	79.86	82.96
Overall	74.32	74.02

Table 1: Average performance compared to the top-8 in [9]

We want to remark that our system uses only the official training and testing data, without including background knowledge of any type. Some of the top-8 systems used background knowledge in order to assist in resolving ambiguities.

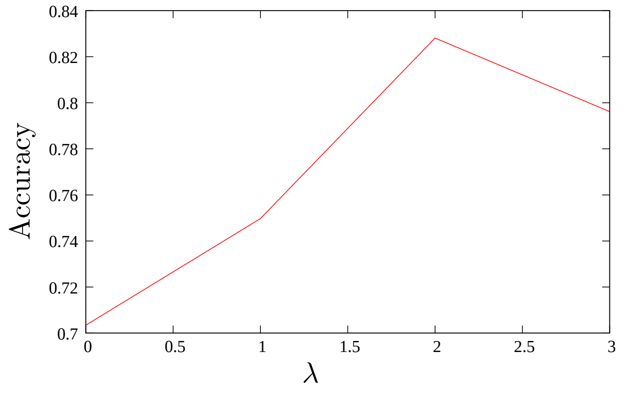


Figure 2: Average accuracy related to parameter $\lambda = 0, 1, 2, 3$

An analysis of the parameter optimization performed in the tuning phase lead us to observe that there is a relation between the dimensionality reduction level applied by SVD and the accuracy achieved for a word disambiguation (see Fig. 2). Words with more than 500 cases in the training dataset were not depicted in the figure because an additional dimension reduction was applied to them (see section 3.1). The graphic in Fig. 2 suggests that

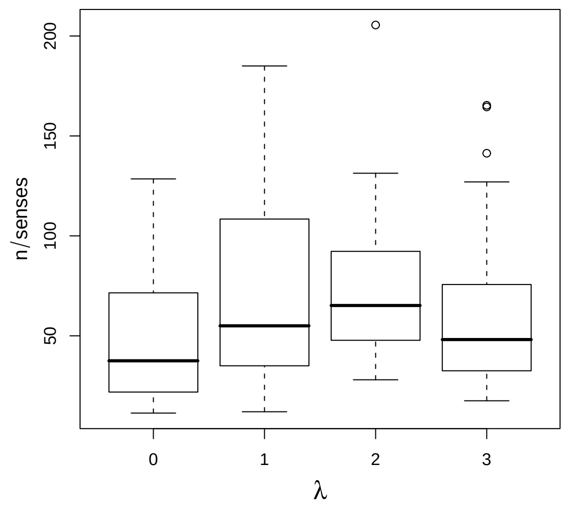


Figure 3: Complexity related to parameter $\lambda = 0, 1, 2, 3$

a dimensionality reduction of half of the features, $\lambda = 2$, is appropriate for words where a high level of accuracy is reached.

In order to analyze the adequacy of the parameter tuning performed, we created a new variable dividing the case number n of the training database by the number of senses for each word. This calculus is meant to represent the complexity of each word. In Fig. 3 the interquartile relationships found among the parameter λ and the complexity of the words is presented. For each value of λ the segments represent the minimum and the maximum value of the complexity, while the bold line shows the median and the rectangular area represents the density of the second and third quartiles. As it can be seen, the evolution of the median value, as well as the minimum values, are similar to the observed in the accuracies. This allows to say that the λ value was properly selected by the automatic selection used, and also that higher values of λ would not ensure better solutions for the most complex words.

5 Conclusions and Future Work

The good results achieved by our system show that the construction of multiclassifiers, together with the use of Bayesian voting to combine word-sense label predictions, plays an important role in disambiguation tasks. The use of the SVD technique in order to reduce the vector representation of cases has been proved to behave appropriately.

We also want to remark that, our disambiguation system has been adapted to the task of disambiguating each one of the 100 words by applying a methodological parameter tuning directed to find the optimal values for each word. This makes possible to have a unique disambiguation system applicable to words with very different characteristics.

Moreover, in our experiments we used only the training data supplied for sense disambiguation in test set, with no inclusion of background knowledge at all, while most of the top-8 systems participating in the task do use some kind of background knowledge. As future work, we intend to make use of such knowledge and hope that results will increase. We also intend to apply this approach to other disambiguation tasks.

References

[1] E. Agirre and O. Lopez de Lacalle. Ubc-alm: Combining k-nn with svd for wsd. In *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval-2007*, pages 342–345, 2007.

[2] M. Berry and M. Browne. *Understanding Search Engines: Mathematical Modeling and Text Retrieval.* SIAM Society for Industrial and Applied Mathematics, ISBN: 0-89871-437-0, Philadelphia, 1999.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] B. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Recognition Classification Techniques.* IEEE Computer Society Press, 1991.

[5] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[6] T. Dietterich. Machine learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.

[7] S. Dumais. Latent semantic analysis. In *ARIST (Annual Review of Information Science Technology)*, volume 38, pages 189–230, 2004.

[8] T. Ho, J. Hull, and S. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[9] S. Pradhan, E. Loper, D. Dligach, and M. Palmer. Semeval-2007 task 17: English lexical sample, srl and all words. In A. for Computational Linguistics, editor, *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval-2007*, pages 87–92, 2007.

[10] A. Zelaia, O. Arregi, and B. Sierra. Ubc-zas: A k-nn based multiclassifier system to perform wsd in a reduced dimensional vector space. In *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval-2007*, pages 358–361, 2007.