Exploring feature spaces with svd and unlabeled data for Word Sense Disambiguation

Eneko Agirre
IXA NLP Group
Univ. of the Basque Country
Donostia, 20018
e.agirre@ehu.es

Oier Lopez de Lacalle
IXA NLP Group
Univ. of the Basque Country
Donostia, 20018
jibloleo@si.ehu.es

David Martínez
IXA NLP Group
Univ. of the Basque Country
Donostia, 20018
davidm@si.ehu.es

Abstract

Current Word Sense Disambiguation systems suffer from the lack of hand-tagged data, as well as performance degradation when moving to other domains. In this paper we explore three different improvements to state-of-the-art systems: 1) using Singular Value Decomposition in order to find correlations among features, trying to deal with sparsity, 2) using unlabeled data from a corpus related to the evaluation corpus, and 3) splitting the feature space into smaller, more coherent, sets. Each of the proposals improves the results, and properly combined they achieve the best results to date for the Senseval 3 lexical sample dataset. The analysis of the results provides further insights and possibilities for the future.

1 Introduction

Many current Natural Language Processing (NLP) systems rely on linguistic knowledge acquired from tagged text via Machine Learning (ML) methods. Statistical or alternative models are learned, and then applied to running text. The main problem faced by such systems is the sparse data problem, due to the small amount of training examples. Focusing on Word Sense Disambiguation (WSD), only a handful of occurrences with sense tags are available per word. For example, if we take the word *channel*, we see that it occurs 5 times in SemCor (Miller *et al.* 93), the only all-words sense-tagged corpus publicly available: the first sense has four occurrences, the second a single occurrence, and the other 5 senses are not represented. For a few words, more extensive training data exists: The Lexical Sample task of Senseval-2 (Edmonds & Cotton 01) provides 145 occurrences for *channel*, but still some of the senses are represented by only 3 or 5 occurrences.

In addition to the sparse data problem, supervised WSD systems are usually trained and tested in texts coming from the same corpus. When training and testing instances come from distinct sources with domain or genre differences, the performance typically drops accordingly (Martínez & Agirre 00).

The impact of the above problems (sparsity and domain shifts) is exemplified by the frustrating handful of systems which are able to beat the simple Most Frequent Sense baseline in the realistic all-words task in both Senseval-2 and Senseval-3 (Snyder & Palmer 04). In these exercises the best systems were trained over SemCor, and the test texts came from The Wall Street Journal and the Brown corpus.

One possible solution to the above problems is to use unlabeled data and appropriate learning techniques that can take advantage of them. Unlabeled data might alleviate the need of hand-labeled data, and, in addition help to adapt the system to new domains. Recently, there have been several attempts in the WSD literature which use co-training (Mihalcea 04) and Principal Component Analysis (Su *et al.* 04). The results have been mixed, with some improvements over baseline supervised systems, but still below the best purely supervised system in the Senseval lexical sample tasks. An exception is (Gliozzo *et al.* 05), which improves the best Senseval-3 results using a combination of kernels and domain information modeled with Singular Value Decomposition (SVD). This last system is closely related to ours, and we will highlight the differences in the related work section.

Alternatively, there is also the preoccupation about the best way to apply ML techniques to supervised settings. The first issue is to represent the context with appropriate features. The last Senseval exercises show that the more feature types one throws into the algorithm, the better are the results (Agirre & Martínez 04). Still, it is not clear which is the best way to profit from the very rich feature space. Apart from the sparsity problem already mentioned, large feature spaces tend to have highly redundant and heterogeneous features (see Section 2.2). As a potential solution, we interpret that SVD (cf. Section 3.1) collapses similar features (i.e. having similar distributions), and will thus be helpful against sparsity and redundancy. Regarding heterogeneity, splitting the feature space might allow the learning algorithm to better capture the patterns in the data.

In this paper we explore three different ways to improve feature modeling:

- Using SVD in order to find correlations among features, trying to deal with sparsity.
- Using unlabeled data from a corpus related to the evaluation corpus coupled with SVD as above.
- Splitting the feature space into smaller, more coherent, sets, trying to better model the feature space.

These improvements need to be combined with state-of-the-art ML algorithms. The methods based on the spatial representation of features (such as Support Vector Machines, Vector Space Models and k-Nearest Neighbors) seem to be the best performing, and we have focused on them (cf. Section 2.3)

We will show that each of the modifications in the feature space improves the results, and properly combined they achieve the best results to date for the Senseval 3 lexical sample dataset. The analysis of the results will provide further insights and possibilities for the future.

The paper is structured as follows. Section 2 reviews the experimental setting and state-of-the-art WSD systems that we used as baselines, including the feature set and ML methods used. Section 3 introduces the improvements proposed in this paper. Section 4 presents the results of these improvements. Section 5 introduces the combination method and its results. Section 6 presents the discussion and related work. Finally, Section 7 draws the conclusions and the future work.

2 Experimental setting and baseline systems

In order to organize the experiments we started building state-of-the-art WSD systems based on our previous experience (Agirre & Martínez 04). In the next sections we will present briefly the main components of the WSD system, that is, the features used to represent the context and the ML algorithms applied. But we first describe the target WSD task and the evaluation methodology.

2.1 Corpus and evaluation

The experiments have been performed using the Senseval-3 English Lexical-Sample data (Mihalcea *et al.* 04). The source corpora was the BNC (Leech 92). WordNet 1.7.1. (Fellbaum 98) was chosen as the sense inventory for nouns and adjectives, while the verb senses came from the *Wordsmyth* dictionary[1]. 57 words (nouns, verbs, and adjectives) were tagged, with 7,860 instances for training and 3,944 for testing.

For the development and fine-tuning of our systems, we have used 3-fold cross validation over the training set, where the three folds were built following stratified sampling. The final evaluation and the comparison with other systems was made over the testing set. The usual precision and recall figures were computed for each system. In all the cases reported here coverage was 100% and precision equalled recall, so we use recall in all tables.

2.2 Features

The feature types can be grouped in three main sets:

Local collocations: bigrams and trigrams formed with the words around the target. These features are constituted by lemmas, word-forms, or PoS tags[2]. Other local features are those formed with the previous/posterior lemma/word-form in the context.

Syntactic dependencies: syntactic dependencies were extracted using heuristic patterns, and regular expressions defined with the PoS tags around the target[3]. The following relations were used: object, subject, noun-modifier, preposition, and sibling.

Bag-of-words features: we extract the lemmas of the content words in the whole context, and in a ±4-word window around the target. We also obtain salient bigrams in the context, with the methods and the software described in (Pedersen 01).

2.3 ML methods

Given an occurrence of a word, the ML methods below return a weight for each sense ($weight(s_k)$). The sense with maximum weight will be selected.

[1]http://www.wordsmyth.net/

[2]The PoS tagging was performed with the fnTBL toolkit (Ngai & Florian 01).

[3]This software was kindly provided by David Yarowsky's group, from the Johns Hopkins University.

Each occurrence or instance is represented by the features found in the context (f_i).

For the **Vector Space Model** (VSM) method, we represent each occurrence context as a vector, where each feature will have a 1 or 0 value to indicate the occurrence/absence of the feature. For each sense in training, one centroid vector is obtained ($\vec{C_{s_k}}$). These centroids are compared with the vectors that represent testing examples ($\vec{f}$), by means of the cosine similarity function (eq. (1)). The closest centroid assigns its sense to the testing example.

$$weight(s_k) = cos(\vec{C}_{s_k}, \vec{f}) = \frac{\vec{C}_{s_k}.\vec{f}}{|\vec{C}_{s_k}||\vec{f}|} \quad (1)$$

Regarding **Support Vector Machines** (SVM) we utilized SVM-Light, a public distribution of SVM by (Joachims 99). The weight for each sense is given by the distance to the hyperplane that supports the classes, that is, the sense s_k versus the rest of senses.

The k **Nearest Neighbor** (k-NN) is a memory based learning method (eq. (2)), where the neighbors are the k most similar contexts, represented by feature vectors ($\vec{c_i}$), of the test vector ($\vec{f}$). The similarity among instances is measured by the cosine of their vectors (as in eq. (1)). The test instance is labeled with the sense obtaining the maximum the sum of the weighted vote of the k most similar contexts. The vote is weighted depending on its (neighbor) position in the ordered rank, with the closest being first. Eq. (2) formalizes k-NN, where C_i corresponds to the sense label of the i-th closest neighbor.

$$\arg\max_{S_j} = \sum_{i=1}^{k} \begin{cases} \frac{1}{i} & \text{if } C_i = S_j \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

3 Improvements for feature modeling

This section presents the three improvements that we propose here as solutions to the data sparsity, redundancy and heterogeneity problems. First, we present the use of SVD on the training and test sets. Next, we introduce unlabeled data into the SVD procedure. Finally, we split the feature space into two smaller sets.

3.1 Singular Value Decomposition (SVD)

SVD is a technique to reduce the dimensions of any problem represented by vectors. It has been widely used in Text Categorization, being the basis of Latent Semantic Analysis. SVD reduces the dimensionality of the feature vectors, finding correlations between features, and helping to deal with data sparseness. We will review briefly SVD as we applied it to WSD.

Let $C = \{t_1, t_2, ..., t_n\}$ be a corpus (set of occurrences of target word), where t_i is an instance from the training set. Let $F = \{f_1, f_2, ..., f_m\}$ be the features appeared in C, let $M \ni \mathbf{R}^{m \times n}$ be a feature-by-instance matrix representing C, where $t_{ij} \in M$ is the frequency of feature f_i in instance t_j. Each word in the Lexical Sample has its own M feature-by-instance matrix. Instead of the frequency, one can try more sophisticated weighting schemes, as we will see in Section 4.2.

SVD decomposes the feature-by-instance matrix (M) into the product of three matrices (eq. (3)):

$$M = U\Sigma V^T = \sum_{i=1}^{k=min\{m,n\}} \sigma_i u_i vi^T \quad (3)$$

U and V, row and column matrix, respectively, have orthonormal columns and Σ is a diagonal matrix which contains k eigenvalues in descending order. Note that in WSD problems the number of instances is much lower than the number of features ($n << m$), so k is always equal to the number of instances. By selecting the first p eigenvalues, we reduce the current space to p dimensions, and can thus project the instances (both training and test) to a reduced space. The equation (4) shows how to make this projection, where $\vec{t}^T$ is the transpose of the vector of features corresponding to one occurrence of the target word.

$$\vec{t_p} = \vec{t}^T U_p \Sigma_p^{-1} \quad (4)$$

Once we project all training and testing instances into the reduced space, we can apply any ML algorithm as usual. SVD has been performed with SVDPACK[4] and GTP[5]. VSM and SVM were fed with the results from SVDPACK and k-NN with the results of GTP.

3.2 Singular Value Decomposition with unlabeled data

The sense (label) of an instance is not used in the process of doing SVD. Taking advantage of this, we can use unlabeled data to have a larger

[4]http://www.netlib.org/svdpack
[5]http://wwww.cs.utk.edu/∼lsi

matrix for each word, and hopefully obtain better correlations in the reduced space. We have used the BNC corpus to get large amounts of unlabeled instances, and thus augment the feature-by-instance matrix M from the previous section into M'. In our experiments we have tested different amounts of unlabeled data, trying with 25% or 50% of the occurrences of the word. We call this process **background learning**.

Once we have done the SVD decomposition of M' we obtain the new U' and $\Sigma_p'^{-1}$, we project training and testing instances as in eq. (4) and proceed applying any ML method.

3.3 Splitting feature space

As seen in Section 2.2, WSD uses a high number of heterogeneous features. The methods mentioned in 2.3 are all based on geometrical properties of the feature space. If we split the problem (the whole space of features) into more coherent feature sets, the classification algorithms should find easier its way in such a simple space. We can thus build separate classifiers for each set of features, and hopefully obtain better results.

In order to test this hypothesis we split the features (cf. Section 2.2) in two subsets:

- **Topical features**: Comprising the bag-of-word features.
- **Local features**: Comprising the local collocations and the syntactic dependencies.

4 Preliminary results

In this section we describe the results of the systems presented in the previous sections: we first comment the baseline methods, then some parameter tuning over SVD, and finally the improved algorithms.

4.1 Results of baseline methods

Initially we tried with k-NN, SVM and VSM (section 2.3). VSM has no parameters, but k-NN needs to find an optimal k (number of neighbors) and SVM allows to optimize the "soft margin". We used 3-fold cross-validation on the Senseval-3 Lexical Sample training set. For k-NN we only tried two values: $k = 5$ and $k = 4$. For SVM we used the "soft margin" value obtained in previous experiments.

Table 1 shows the results from cross-validation. We can see that the results of VSM and k-NN are

Classifiers	Recall
k-NN k=5	67.7
k-NN k=4	67.4
SVM	62.3
VSM	68.0

Table 1: Results for baseline classifiers in 3-fold cross-validation (Senseval-3 training set).

Classifiers	Recall
k-NN k=5	70.5
SVM	71.2
VSM	71.5

Table 2: Results for baseline classifiers in the Senseval-3 Lexical Sample test set.

very similar, with VSM outperforming k-NN for 0.3 points, and SVMperforming lower. For the rest of the paper, we set $k = 5$ for all uses of k-NN. The results on the test set are shown in Table 2, with VSM increasing its advantage over k-NN and SVM in the middle of both.

4.2 Parameter setting for SVD

SVD needs to set several parameters which can affect the performance. In order to set those parameters we run several preliminary experiments using SVD coupled with k-NN using 3-fold cross-validation as before. In the rest of the paper, SVD was performed using the following parameters:

- Number of desired **dimensions**: We tried with 100, 200, 300, 500 and 1000 dimensions, and the best performance was obtained with 200 dimensions.
- **Weighting scheme** for the frequencies in the feature-by-instance matrix: We tried different classic schemes, including local weighting formulas such as term frequency (tf), *log* and *binary*, and global measures like *idf* and *entropy*. For this work we have used *log* and *entropy* weighting scheme, replacing $t_{ij} \in M$ (cf. Section 3.1) by $log(t_{ij}) \cdot entropy(i)$.
- **Threshold** for global frequency (g): After building the matrix we can remove features that are very common (the less informative). We tried with different thresholds, and finally we chose to accept all features ($g = 0$).

Classifiers	Recall
k-NN k=5	67.7
SVM	62.3
VSM	68.0
k-NN-SVD k=5	69.8
SVM-SVD	61.2
VSM-SVD	63.9

Table 3: Results for k-NN and VSM with SVD in 3-fold cross-validation (Senseval-3 training set).

k-NN ($k = 5$)	Recall	diff.
plain	67.7	—
local+topical	69.4	+1.7
SVD	69.6	+1.9
SVD (25% BNC)	69.2	+1.5
SVD (50% BNC)	69.6	+1.9

Table 4: Improved k-NN classifier in 3-fold cross-validation (Senseval-3 training set). Plain stands for baseline k-NN.

Classifiers	Recall	diff.
plain	70.5	—
local+topical	70.8	+0.3
SVD	70.7	+0.2
SVD (25% BNC)	70.8	+0.2
SVD (50% BNC)	71.2	+0.7
VSM	71.5	+1.0
SVM	71.2	+0.7

Table 5: Improved k-NN classifier in the Senseval-3 Lexical Sample test set. Plain stands for baseline k-NN. VSM and SVM results are also provided for comparison.

4.3 Results of improved systems

In this section, we show how the proposed improvements affect the performance. Table 3 presents the results of doing SVD, and then applying VSM, SVM and k-NN over the reduced space. We can observe that only k-NN improves performance, with VSM and SVM getting lower results. These and other prior experiments motivated us to only use k-NN on the improved systems.

Table 4 shows the results on the training set for the baseline k-NN systems, as well as all improvements explored. The difference over the baseline system shows that all improvements were positive, raising from 1.5 to 1.9 the performance of the baseline. Still, there is no improvement observed when introducing unlabeled data into SVD (25% BNC and 50% BNC in Table 4) compared to using labeled data only (SVD in Table 4).

Table 5 shows the same data for all baseline systems (including VSM and SVM) on the test set. The improvement here is lower but consistent with Table 4. The only difference is that using 25% or 50% of the BNC as unlabeled data for SVD is better than not using labeled data. Table 5 also presents the results of the other two baseline systems, showing that all k-NN systems are below VSM and SVM. This motivated us to try to combine the k-NN classifiers.

5 Combining several k-NN systems

The results from the previous section show that the improved systems (Section 3) are able to increase the results of k-NN, but are still below our SVM and VSM baseline systems. The key observation here is that under each of the improved classifiers there is a slightly different feature space. All of them provide improvements, and are therefore able to generalize interesting properties of the problem space. If we are able to combine them properly, we might be able to further improve the results.

The combination of classifiers is an active area of research. Here we exploited the fact that a k-NN classifier can be seen as k points casting each one vote, making easy a combination of several k-NN classifiers. For instance, if we have two k-NN classifiers of $k = 5$, c_1 and c_2, then we can combine them into a single classifier equivalent to $k = 10$. In order to carry through the properties of each feature space, we decided to weight each vote by the cosine similarity of that point instead of the rank. We need to note that this combination method was also used in the previous section to combine the local and topical classifiers.

Table 6 shows the results over the training set. Plain stands for the baseline k-NN system. The following rows show the improved systems from the previous Section. Then the results of combining the algorithms two by two are shown, where each of the improved systems has been combined with the baseline k-NN system. The results show that all combinations attain better results than any of their components. We can also see that, in this setting, using unlabeled data (plain+SVD with 50%) improves slightly over not using it (plain+SVD). Finally, the full combina-

k-NN($k = 5$)	**Recall**	**diff.**
plain	67.7	—
local+topical	69.4	+1.7
SVD	69.6	+1.9
SVD (25% BNC)	69.2	+1.5
SVD (50% BNC)	69.6	+1.9
plain + local+topical	69.9	+2.2
plain + SVD	70.7	+3.0
plain + SVD (25% BNC)	70.7	+3.0
plain + SVD (50% BNC)	70.8	+3.1
full combination	**71.9**	+4.2

Table 6: Results for different combinations of k-NN classifiers in 3-fold cross-validation (Senseval-3 training set)

Classifiers	**Recall**	**diff.**
plain	70.5	—
local+topical	70.8	+0.3
SVD	70.7	+0.2
SVD(%25 BNC)	70.8	+0.2
SVD(%50 BNC)	71.2	+0.7
plain + local+topical	71.5	+1.0
plain + SVD	71.2	+0.7
plain + SVD (25% BNC)	72.3	+1.8
plain + SVD (50% BNC)	72.7	+2.2
full combination	**73.4**	+2.9
SVM	71.2	—
VSM	71.5	—
Best S3	72.9	—
(Gliozzo *et al.* 05)	73.3	—

Table 7: Results for different combinations of k-NN classifiers in the Senseval-3 Lexical Sample test set. Plain stands for baseline k-NN. VSM and SVM results are also provided, as well as the best Senseval-3 system and the best result published to date.

tion of all 5 systems provides the best results. Note that for the full combination, we applied SVD (with only labeled data, plus 25% of BNC and 50% of BNC) also to the local and topical classifiers.

The results on the test set, Table 7, confirm the cross-validation results. Note that unlabeled data makes a more significant improvement over plain+SVD. Below the combined system, Table 7 also shows our baseline systems, as well as the best system in the Senseval 3 competition and the best reported result to date. The full combination of our k-NN systems attains the best results of them all.

6 Discussion and related work

The results show that we have been able to better model the feature space. SVD helps to find correlations among the features, and thus alleviate the sparse data and redundancy problems. Including unlabeled data provides very narrow performance increases, but combined with the other classifiers it makes a difference. Splitting the feature space in two and combining the two spaces also improves the results. These improvements in isolation are not very large. In fact, the resulting k-NN systems are below our SVM and SVM baseline systems for the original feature set. But when we combine the k-NN algorithms over each of the feature spaces, we attain the best results to date in the Senseval-3 dataset.

We think that the reason explaining the extraordinary performance of the combination is that each of the changes in the feature space helps finding regularities in the data that k-NN could not find before. When we combine each of the simpler k-NN systems, we are looking for the word sense that is closest to the target instance in as many of the changed feature spaces as possible.

Some of the findings in this paper are confirmed in related work, but this paper integrates them in a single task (WSD) and shows that they provide the best performance. For instance, (Kohomban & Lee 05) show in a different WSD task that building separate k-NN classifiers from different subset of features and combining them works better than constructing a single classifier with the entire feature set. In (Gliozzo *et al.* 05), instead of splitting the feature space and then combining the classifiers, they use specialized kernels to model the similarity for each kind of features. They also use SVD but only for bag-of-words features, while we apply SVD to all features. The good performance of coupling k-NN and SVD are well known in the ML literature, e.g. (Thomasian *et al.* 05) on a image retrieval task. (Dietterich 98) says that spliting features only works when the feature space is higly redundant. We already mentioned in the Introduction other works which make use of unlabeled data on a WSD setting.

7 Conclusions and Future Work

In this paper we have explored feature modeling, trying to tackle sparse data, redundancy and heterogeneity in the feature set. We have proposed and evaluated three improvements: 1) using SVD in order to find correlations among features and deal with sparsity and redundancy, 2) using unlabeled data from a corpus related to the evaluation corpus in order to provide background knowledge, and 3) splitting the feature space into smaller, more coherent, sets. Each of the proposals improves the results for a k-NN classifier, and properly combined they provide the best results to date for the Senseval-3 lexical sample dataset.

In the discussion we have argued that this improvements help to model better the feature space, which, coupled with a ML algorithm well suited for combination such as k-NN, explain the good results. This opens new feature modeling possibilities. In particular we are thinking of finer splits of the feature space, using kernels to better model similarity for certain features. On the other hand we have shown that unlabeled data helps, and we would like to better explore which is the situation when the training and test data come from distinct corpora or domains.

Acknowledgements

We wish to thank Basilio Seirra and Ana Zelaia, from the University of Basque Country, for helping us with the SVD and k-NN methods. This research has been partially founded by the European Commision (MEANING IST-2001-34460), Ministry of Education and Science (CESS-ECE HUM2004-21127-E) and the Basque Goverment (EU-SEMCOR S-PE04UN10).

References

(Agirre & Martínez 04) E. Agirre and D. Martínez. Smoothing and Word Sense Disambiguation. In *Proceedings of EsTAL - España for Natural Language Processing*, Alicante, Spain, 2004.

(Dietterich 98) Thomas G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.

(Edmonds & Cotton 01) P. Edmonds and S. Cotton. SENSEVAL-2: Overview. In *Proceedings of the Second International Workshop on evaluating Word Sense Disambiguation Systems.*, Toulouse, France, 2001.

(Fellbaum 98) C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

(Gliozzo *et al.* 05) Alfio Massimiliano Gliozzo, Claudio Giuliano, and Carlo Strapparava. Domain Kernels for Word Sense Disambiguation. *43nd Annual Meeting of the Association for Computational Linguistics. (ACL-05)*, 2005.

(Joachims 99) T. Joachims. Making Large–Scale SVM Learning Practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

(Kohomban & Lee 05) Upali S. Kohomban and Wee S. Lee. Learning Semantic Classes for Word Sense Disambiguation. In *43nd Annual Meeting of the Association for Computational Linguistics. (ACL-05)*, University of Michigan, Ann Arbor, 2005.

(Leech 92) G. Leech. 100 million words of English: the British National Corpus. *Languaje Research*, 28(1):1–13, 1992.

(Martínez & Agirre 00) David Martínez and Eneko Agirre. One Sense per Collocation and Genre/Topic Variations. *Conference on Empirical Method in Natural Language*, 2000.

(Mihalcea 04) Rada Mihalcea. Co-training and Self-training for Word Sense Disambiguation. In *In Proceedings of the Conference on Natural Language Learning (CoNLL 2004)*, Boston, USA, 2004.

(Mihalcea *et al.* 04) R. Mihalcea, T. Chklovski, and Adam Killgariff. The Senseval-3 English lexical sample task. In *Proceedings of the 3rd ACL workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL)*, Barcelona, Spain, 2004.

(Miller *et al.* 93) G.A. Miller, C. Leacock, R. Tengi, and R.Bunker. A Semantic Concordance. In *Proceedings of the ARPA Human Language Technology Workshop. Distributed as* Human Language Technology *by San Mateo, CA: Morgan Kaufmann Publishers.*, pages 303–308, Princeton, NJ, 1993.

(Ngai & Florian 01) G. Ngai and R. Florian. Transformation-Based Learning in the Fast Lane. *Proceedings of the Second Conference of the North American Chapter of the Association for Computational Linguistics, pages 40-47, Pittsburgh, PA, USA*, 2001.

(Pedersen 01) T. Pedersen. A Decision Tree of Bigrams is an Accurate Predictor of Word Sense. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-01)*, Pittsburgh, PA, 2001.

(Snyder & Palmer 04) B. Snyder and M. Palmer. The English all-words task. In *Proceedings of the 3rd ACL workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL)*, Barcelona, Spain, 2004.

(Su *et al.* 04) Weifeng Su, Dekai Wu, and Marine Carpuat. Semi-Supervised Training of a Kernel PCA-Based Model for Word Sense Disambiguation. *20th International Conference on Computational Linguistics (COLING-2004)*, 2004.

(Thomasian *et al.* 05) A. Thomasian, Y. Li, and L. Zhang. Exact k-NN queries on clustered SVD datasets. *Information Processing Letters (ILP)*, 94:247–252, 2005.