A Comparison between Supervised Learning Algorithms for Word Sense Disambiguation*

Gerard Escudero, Lluís Màrquez, and **German Rigau**
TALP Research Center. LSI Department. Universitat Politècnica de Catalunya (UPC)
Jordi Girona Salgado 1–3. E-08034 Barcelona. Catalonia
{escudero,lluism,g.rigau}@lsi.upc.es

Abstract

This paper describes a set of comparative experiments, including cross–corpus evaluation, between five alternative algorithms for supervised Word Sense Disambiguation (WSD), namely Naive Bayes, Exemplar-based learning, *SNoW*, Decision Lists, and Boosting. Two main conclusions can be drawn: 1) The LazyBoosting algorithm outperforms the other four state-of-the-art algorithms in terms of accuracy and ability to tune to new domains; 2) The domain dependence of WSD systems seems very strong and suggests that some kind of adaptation or tuning is required for cross–corpus application.

1 Introduction

Word Sense Disambiguation (WSD) is the problem of assigning the appropriate meaning (or sense) to a given word in a text or discourse. Resolving the ambiguity of words is a central problem for large scale language understanding applications and their associate tasks (Ide and Véronis, 1998). Besides, WSD is one of the most important open problems in NLP. Despite the wide range of approaches investigated (Kilgarriff and Rosenzweig, 2000) and the large effort devoted to tackling this problem, to date, no large-scale broad-coverage and highly accurate WSD system has been built.

One of the most successful current lines of research is the corpus-based approach, in which statistical or Machine Learning (ML) algorithms have been applied to learn statistical models or classifiers from corpora in order to perform WSD. Generally, supervised approaches (those that learn from previously semantically annotated corpora) have obtained better results than unsupervised methods on small sets of selected ambiguous words, or artificial pseudowords. Many standard ML algorithms for supervised learning have been applied, such as: Decision Lists (Yarowsky, 1994; Agirre and Martínez, 2000), Neural Networks (Towell and Voorhees, 1998), Bayesian learning (Bruce and Wiebe, 1999), Exemplar-based learning (Ng, 1997), Boosting (Escudero et al., 2000a), etc. Further, in (Mooney, 1996) some of the previous methods are compared jointly with Decision Trees and Rule Induction algorithms, on a very restricted domain.

Although some published works include the comparison between some alternative algorithms (Mooney, 1996; Ng, 1997; Escudero et al., 2000a; Escudero et al., 2000b), none of them addresses the issue of the portability of supervised ML algorithms for WSD, i.e., testing whether the accuracy of a system trained on a certain corpus can be extrapolated to other corpora or not. We think that the study of the domain dependence of WSD —in the style of other studies devoted to parsing (Sekine, 1997; Ratnaparkhi, 1999)— is needed to assure the validity of the supervised approach, and to determine to which extent a tuning pre–process is necessary to make real WSD systems portable. In this direction, this work compares five different ML algorithms and explores their portability and tuning ability by training and testing them on different corpora.

2 Learning Algorithms Tested

Naive-Bayes (NB). Naive Bayes is intended as a simple representative of statistical learning methods. It has been used in its most classi-

* This research has been partially funded by the Spanish Research Department (CICYT's project TIC98–0423–C06), by the EU Commission (NAMIC IST-1999-12392), and by the Catalan Research Department (CIRIT's consolidated research group 1999SGR-150 and CIRIT's grant 1999FI 00773).

cal setting (Duda and Hart, 1973). That is, assuming the independence of features, it classifies a new example by assigning the class that maximizes the conditional probability of the class given the observed sequence of features of that example. Model probabilities are estimated during the training process using relative frequencies. To avoid the effect of zero counts, a very simple smoothing technique has been used, which was proposed in (Ng, 1997).

Despite its simplicity, Naive Bayes is claimed to obtain state–of–the–art accuracy on supervised WSD in many papers (Mooney, 1996; Ng, 1997; Leacock et al., 1998).

Exemplar-based Classifier (EB). In exemplar, instance, or memory–based learning (Aha et al., 1991) no generalization of training examples is performed. Instead, the examples are simply stored in memory and the classification of new examples is based on the most similar stored exemplars. In our implementation, all examples are kept in memory and the classification is based on a k–NN (Nearest–Neighbours) algorithm using Hamming distance to measure closeness. For k's greater than 1, the resulting sense is the weighted majority sense of the k nearest neighbours —where each example votes its sense with a strength proportional to its closeness to the test example.

Exemplar–based learning is said to be the best option for WSD (Ng, 1997). Other authors (Daelemans et al., 1999) point out that exemplar–based methods tend to be superior in language learning problems because they do not forget exceptions.

The SNoW Architecture (SN). *SNoW* is a Sparse Network of linear separators which utilizes the Winnow learning algorithm[1]. In the *SNoW* architecture there is a winnow node for each class, which learns to separate that class from all the rest. During training, which is performed in an on–line fashion, each example is considered a positive example for the winnow node associated to its class and a negative example for all the others. A key point that allows a fast learning is that the winnow nodes are not connected to all features but only to those that are "relevant" for their class. When classifying a new example, *SNoW* is similar to a neural network which takes the input features and outputs the class with the highest activation. Our implementation of *SNoW* for WSD is explained in (Escudero et al., 2000c).

SNoW is proven to perform very well in high dimensional NLP problems, where both the training examples and the target function reside very sparsely in the feature space (Roth, 1998), e.g: context–sensitive spelling correction, POS tagging, PP–attachment disambiguation, etc.

Decision Lists (DL). In this setting, a Decision List is a list of features extracted from the training examples and sorted by a log–likelihood measure. This measure estimates how strong a particular feature is as an indicator of a specific sense (Yarowsky, 1994). When testing, the decision list is checked in order and the feature with the highest weight that matches the test example is used to select the winning word sense. Thus, only the single most reliable piece of evidence is used to perform disambiguation. Regarding the details of implementation (smoothing, pruning of the decision list, etc.) we have followed (Agirre and Martínez, 2000).

Decision Lists were one of the most successful systems on the 1st Senseval competition for WSD (Kilgarriff and Rosenzweig, 2000).

LazyBoosting (LB). The main idea of boosting algorithms is to combine many simple and moderately accurate hypotheses (*weak classifiers*) into a single, highly accurate classifier. The weak classifiers are trained sequentially and, conceptually, each of them is trained on the examples which were most difficult to classify by the preceding weak classifiers. These weak hypotheses are then linearly combined into a single rule called the *combined hypothesis*.

Schapire and Singer's real AdaBoost.MH algorithm for multiclass multi–label classification (**?**) has been used. It constructs a combination of very simple weak hypotheses that test the value of a single boolean predicate and make a real–valued prediction based on that value. LazyBoosting (Escudero et al., 2000a) is a simple modification of the AdaBoost.MH algorithm, which consists in reducing the feature space that is explored when learning each weak classifier. This modification significantly increases the efficiency of the learning process

[1]The Winnow algorithm (Littlestone, 1988) consists of a linear threshold algorithm with multiplicative weight updating for 2-class problems.

with no loss in accuracy.

3 Setting

A number of comparative experiments has been carried out on a subset of 21 highly ambiguous words of the DSO corpus, which is a semantically annotated English corpus collected by Ng and colleagues (Ng and Lee, 1996). Each word is treated as a different classification problem. The 21 words comprise 13 nouns (age, art, body, car, child, cost, head, interest, line, point, state, thing, work) and 8 verbs (become, fall, grow, lose, set, speak, strike, tell), which frequently appear in the WSD literature. The average number of senses per word is close to 10 and the number of training examples is around 1,000.

The DSO corpus contains sentences from two different corpora, namely Wall Street Journal (WSJ) and Brown Corpus (BC). Therefore, it is easy to perform experiments about the portability of systems by training them on the WSJ part (A part, hereinafter) and testing them on the BC part (B part, hereinafter), or vice-versa.

Two kinds of information are used to train classifiers: *local* and *topical* context. Let "$\ldots w_{-3}\ w_{-2}\ w_{-1}\ w\ w_{+1}\ w_{+2}\ w_{+3} \ldots$" be the context of consecutive words around the word w to be disambiguated, and $p_{\pm i}$ ($-3 \leq i \leq 3$) be the part–of–speech tag of word $w_{\pm i}$. Attributes referring to local context are the following 15: p_{-3}, p_{-2}, p_{-1}, p_{+1}, p_{+2}, p_{+3}, w_{-1}, w_{+1}, (w_{-2}, w_{-1}), (w_{-1}, w_{+1}), (w_{+1}, w_{+2}), (w_{-3}, w_{-2}, w_{-1}), (w_{-2}, w_{-1}, w_{+1}), (w_{-1}, w_{+1}, w_{+2}), and (w_{+1}, w_{+2}, w_{+3}), where the last seven correspond to collocations of two and three consecutive words. The topical context is formed by $c_1, \ldots, c_m$, which stand for the unordered set of open class words appearing in the sentence[2]. Details about how the different algorithms translate this information into features can be found in (Escudero et al., 2000c).

4 Comparing the five approaches

The five algorithms, jointly with a naive Most-Frequent-sense Classifier (MFC), have been tested, by 10–fold cross validation, on 7 different combinations of training–test sets[3]. Accuracy figures, micro–averaged over the 21 words and over the ten folds, are reported in table 1. The comparison leads to the following conclusions:

As expected, the five algorithms significantly outperform the baseline MFC classifier. Among them, three groups can be observed: NB, DL, and SN perform similarly; LB outperforms all the other algorithms in all experiments; and EB is somewhere in between. The difference between LB and the rest is statistically significant in all cases except when comparing LB to the EB approach in the case marked with an asterisk[4].

Extremely poor results are observed when testing the portability of the systems. Restricting to LB results, it can be observed that the accuracy obtained in A–B is 47.1%, while the accuracy in B–B (which can be considered an upper bound for LB in B corpus) is 59.0%, that is, that there is a difference of 12 points. Furthermore, 47.1% is only slightly better than the most frequent sense in corpus B, 45.5%.

Apart from accuracy figures, the comparison between the predictions made by the five methods on the test sets provides interesting information about the relative behaviour of the algorithms. Table 2 shows the agreement rates and the Kappa statistics[5] between all pairs of methods in the A+B–A+B experiment. Note that 'DSO' stands for the annotation of DSO corpus, which is taken as the correct one.

It can be observed that NB obtains the most similar results with regard to MFC in agreement and Kappa values. The agreement ratio is 74%, that is, almost 3 out of 4 times it predicts the most frequent sense. On the other extreme, LB obtains the most similar results with regard to DSO in agreement and Kappa values, and it has the least similar with regard to MFC, suggesting

[2]This set of attributes corresponds to that used in (Ng and Lee, 1996), with the exception of the morphology of the target word and the verb–object syntactic relation.

[3]The combinations of training–test sets are called: A+B–A+B, A+B–A, A+B–B, A–A, B–B, A–B, and B–A, respectively. In this notation, the training set is placed on the left hand side of symbol "–", while the test set is on the right hand side. For instance, A–B means that the training set is corpus A and the test set is corpus B. The symbol "+" stands for set union.

[4]Statistical tests of significance applied: McNemar's test and 10-fold cross-validation paired Student's t-test at a confidence value of 95% (Dietterich, 1998).

[5]The Kappa statistic (Cohen, 1960) is a better measure of inter–annotator agreement which reduces the effect of chance agreement. It has been used for measuring inter–annotator agreement during the construction of semantic annotated corpora (Véronis, 1998; Ng et al., 1999). A Kappa value of 1 indicates perfect agreement, while 0.8 is considered as indicating good agreement.

	Accuracy (%)						
	A+B–A+B	A+B–A	A+B–B	A–A	B–B	A–B	B–A
MFC	46.55±0.71	53.90±2.01	39.21±1.90	55.94±1.10	45.52±1.27	36.40	38.71
Naive Bayes	61.55±1.04	67.25±1.07	55.85±1.81	65.86±1.11	56.80±1.12	41.38	47.66
Decision Lists	61.58±0.98	67.64±0.94	55.53±1.85	67.57±1.44	56.56±1.59	43.01	48.83
SNoW	60.92±1.09	65.57±1.33	56.28±1.10	67.12±1.16	56.13±1.23	44.07	49.76
Exemplar–based	63.01±0.93	69.08±1.66	56.97±1.22	68.98±1.06	57.36±1.68	45.32	51.13
LazyBoosting	**66.32**±1.34	**71.79**±1.51	**60.85**±1.81	**71.26**±1.15	**58.96**±1.86	**47.10**	**51.99***

Table 1: Accuracy results (± standard deviation) of the methods on all training–test combinations

	A+B–A+B						
	DSO	MFC	NB	EB	SN	DL	LB
DSO	—	46.6	61.6	63.0	60.9	61.6	66.3
MFC	-0.19	—	73.9	60.0	55.9	64.9	54.9
NB	0.24	-0.09	—	76.3	74.5	76.8	71.4
EB	0.36	-0.15	0.44	—	69.6	70.7	72.5
SN	0.36	-0.17	0.44	0.44	—	67.5	69.0
DL	0.32	-0.13	0.40	0.41	0.38	—	69.9
LB	0.44	-0.17	0.37	0.50	0.46	0.42	—

Table 2: Kappa statistic (below diagonal) and % of agreement (above diagonal) between all methods in the A+B–A+B experiment

that LB is the algorithm that better learns the behaviour of the DSO examples.

In absolute terms, the Kappa values are very low. But, as it is suggested in (Véronis, 1998), evaluation measures should be computed relative to the agreement between the human annotators of the corpus and not to a theoretical 100%. It seems pointless to expect more agreement between the system and the reference corpus than between the annotators themselves. Contrary to the intuition that the agreement between human annotators should be very high in the WSD task, some papers report surprisingly low figures. For instance, (Ng et al., 1999) reports an accuracy rate of 56.7% and a Kappa value of 0.317 when comparing the annotation of a subset of the DSO corpus performed by two independent research groups. From this perspective, the Kappa value of 0.44 achieved by LB in A+B–A+B could be considered an excellent result. Unfortunately, the subset of the DSO corpus studied by (Ng et al., 1999) and that used in this report are not the same and, thus, a direct comparison is not possible.

4.1 About the tuning to new domains

This experiment explores the effect of a simple tuning process consisting in adding to the original training set A a relatively small sample of manually sense–tagged examples of the new domain B. The size of this supervised portion varies from 10% to 50% of the available corpus in steps of 10% (the remaining 50% is kept for testing)[6]. This experiment will be referred to as A+%B–B[7]. In order to determine to which extent the original training set contributes to accurately disambiguating in the new domain, we also calculate the results for %B–B, that is, using only the tuning corpus for training.

Figure 1 graphically presents the results obtained by all methods. Each plot contains the A+%B–B and %B–B curves, and the straight lines corresponding to the lower bound MFC, and to the upper bounds B–B and A+B–B.

As expected, the accuracy of all methods grows (towards the upper bound) as more tuning corpus is added to the training set. However, the relation between A+%B–B and %B–B reveals some interesting facts. In plots (c) and (d), the contribution of the original training corpus is null, while in plots (a) and (b), a degradation on the accuracy is observed. Summarizing, these results suggest that for NB, DL, SN, and EB methods it is not worth keeping the original training examples. Instead, a better (but disappointing) strategy would be simply using the tuning corpus. However, this is not the situation of LB —plot (d)— for which a moderate, but consistent, improvement of accuracy is observed when retaining the original training set.

[6] Tuning examples can be weighted more highly than the training examples to force the learning algorithm to adapt more quickly to the new corpus. Some experiments in this direction revealed that slightly better results can be obtained, though the improvement was not statistically significant.

[7] The converse experiment B+%A–A is not reported in this paper due to space limitations. Results can be found in (Escudero et al., 2000c).

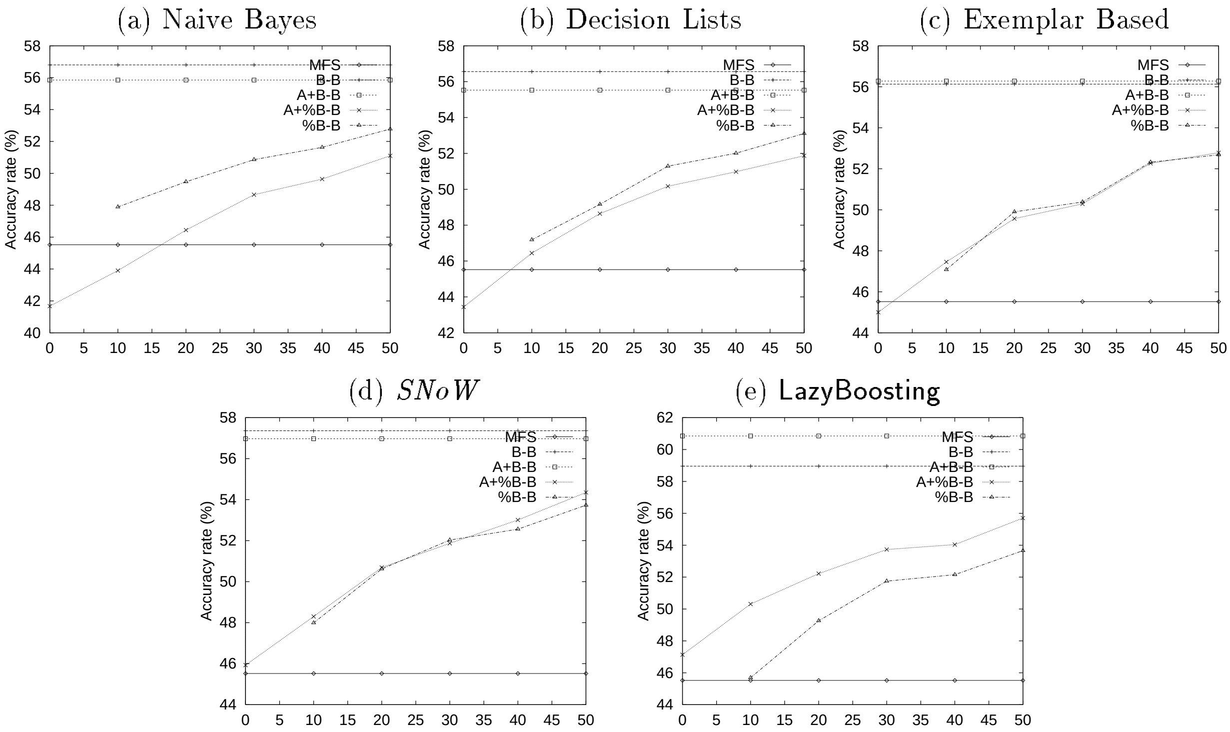

Figure 1: Results of the tuning experiment

We observed that part of the poor results obtained is explained by: 1) corpus A and B have a very different distribution of senses, and, therefore, different a–priori biases; furthermore, 2) examples of corpus A and B contain different information and, therefore, the learning algorithms acquire different (and non interchangeable) classification clues from both corpora. The study of the rules acquired by LazyBoosting from WSJ and BC helped understanding the differences between corpora. On the one hand, the type of features used in the rules was significantly different between corpora and, additionally, there were very few rules that applied to both sets. On the other hand, the sign of the prediction of many of these common rules was somewhat contradictory between corpora. See (Escudero et al., 2000c) for details.

4.2 About the training data quality

The observation of the rules acquired by LazyBoosting could also help improving data quality in a semi–supervised fashion. It is known that mislabelled examples resulting from annotation errors tend to be hard examples to classify correctly and, therefore, tend to have large weights in the final distribution. This observation allows both to identify the noisy examples and use LazyBoosting as a way to improve the training corpus.

A preliminary experiment has been carried out in this direction by studying the rules acquired by LazyBoosting from the training examples of the word *state*. The manual revision, by four different people, of the 50 highest scored rules, allowed us to identify 28 noisy training examples. 11 of them were clear tagging errors, and the remaining 17 were not coherently tagged and very difficult to judge, since the four annotators achieved systematic disagreement (probably due to the extremely fine grained sense definitions involved in these examples).

5 Conclusions

This work reports a comparative study of five ML algorithms for WSD, and provides some results on cross corpora evaluation and domain re-tuning.

Regarding portability, it seems that the performance of supervised sense taggers is not guaranteed when moving from one domain to another (e.g. from a balanced corpus, such as BC, to an economic domain, such as WSJ).

These results imply that some kind of adaptation is required for cross–corpus application. Consequently, it is our belief that a number of issues regarding portability, tuning, knowledge acquisition, etc., should be thoroughly studied before stating that the supervised ML paradigm is able to resolve a realistic WSD problem.

Regarding the ML algorithms tested, LazyBoosting emerges as the best option, since it outperforms the other four state-of-the-art methods in all experiments. Furthermore, this algorithm shows better properties when tuned to new domains. Future work is planned for an extensive evaluation of LazyBoosting on the WSD task. This would include taking into account additional/alternative attributes, learning curves, testing the algorithm on other corpora, etc.

References

E. Agirre and D. Martínez. 2000. Decision Lists and Automatic Word Sense Disambiguation. In *Proceedings of the COLING Workshop on Semantic Annotation and Intelligent Content*.

D. Aha, D. Kibler, and M. Albert. 1991. Instance–based Learning Algorithms. *Machine Learning*, 7:37–66.

R. F. Bruce and J. M. Wiebe. 1999. Decomposable Modeling in Natural Language Processing. *Computational Linguistics*, 25(2):195–207.

J. Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Journal of Educational and Psychological Measurement*, 20:37–46.

W. Daelemans, A. van den Bosch, and J. Zavrel. 1999. Forgetting Exceptions is Harmful in Language Learning. *Machine Learning*, 34:11–41.

T. G. Dietterich. 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7).

R. O. Duda and P. E. Hart. 1973. *Pattern Classification and Scene Analysis*. Wiley & Sons.

G. Escudero, L. Màrquez, and G. Rigau. 2000a. Boosting Applied to Word Sense Disambiguation. In *Proceedings of the 12th European Conference on Machine Learning, ECML*.

G. Escudero, L. Màrquez, and G. Rigau. 2000b. Naive Bayes and Exemplar-Based Approaches to Word Sense Disambiguation Revisited. In *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI*.

G. Escudero, L. Màrquez, and G. Rigau. 2000c. On the Portability and Tuning of Supervised Word Sense Disambiguation Systems. Research Report LSI-00-30-R, Software Department (LSI). Technical University of Catalonia (UPC).

N. Ide and J. Véronis. 1998. Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, 24(1):1–40.

A. Kilgarriff and J. Rosenzweig. 2000. English SENSEVAL: Report and Results. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation, LREC*.

C. Leacock, M. Chodorow, and G. A. Miller. 1998. Using Corpus Statistics and WordNet Relations for Sense Identification. *Computational Linguistics*, 24(1):147–166.

N. Littlestone. 1988. Learning Quickly when Irrelevant Attributes Abound. *Machine Learning*, 2:285–318.

R. J. Mooney. 1996. Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing, EMNLP*.

H. T. Ng and H. B. Lee. 1996. Integrating Multiple Knowledge Sources to Disambiguate Word Senses: An Exemplar-based Approach. In *Proceedings of the 34th Annual Meeting of the ACL*.

H. T. Ng, C. Lim, and S. Foo. 1999. A Case Study on Inter-Annotator Agreement for Word Sense Disambiguation. In *Procs. of the ACL SIGLEX Workshop: Standardizing Lexical Resources*.

H. T. Ng. 1997. Exemplar-Base Word Sense Disambiguation: Some Recent Improvements. In *Procs. of the 2nd Conference on Empirical Methods in Natural Language Processing, EMNLP*.

A. Ratnaparkhi. 1999. Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning*, 34:151–175.

D. Roth. 1998. Learning to Resolve Natural Language Ambiguities: A Unified Approach. In *Proceedings of the National Conference on Artificial Intelligence, AAAI '98*.

R. E. Schapire and Y. Singer. 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3):297–336.

S. Sekine. 1997. The Domain Dependence of Parsing. In *Proceedings of the 5th Conference on Applied Natural Language Processing, ANLP*.

G. Towell and E. M. Voorhees. 1998. Disambiguating Highly Ambiguous Words. *Computational Linguistics*, 24(1):125–146.

J. Véronis. 1998. A study of polysemy judgements and inter–annotator agreement. In *Programme and advanced papers of the Senseval workshop*, Herstmonceux Castle, England.

D. 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 ACL*.