Boosting Applied to 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. In this paper Schapire and Singer's AdaBoost.MH boosting algorithm is applied to the Word Sense Disambiguation (WSD) problem. Initial experiments on a set of 15 selected polysemous words show that the boosting approach surpasses Naive Bayes and Exemplar–based approaches, which represent state–of–the–art accuracy on supervised WSD. In order to make boosting practical for a real learning domain of thousands of words, several ways of accelerating the algorithm by reducing the feature space are studied. The best variant, which we call LazyBoosting, is tested on the largest sense–tagged corpus available containing 192,800 examples of the 191 most frequent and ambiguous English words. Again, boosting compares favourably to the other benchmark algorithms.

1 Introduction

Word Sense Disambiguation (WSD) is the problem of assigning the appropriate meaning (sense) to a given word in a text or discourse. This meaning is distinguishable from other senses potentially attributable to that word. Resolving the ambiguity of words is a central problem for language understanding applications and their associated tasks [11], including, for instance, machine translation, information retrieval and hypertext navigation, parsing, spelling correction, reference resolution, automatic text summarization, etc.

WSD is one of the most important open problems in the Natural Language Processing (NLP) field. Despite the wide range of approaches investigated and the large effort devoted to tackling this problem, it is a fact that to date no large–scale, broad coverage and highly accurate word sense disambiguation system has been built.

The most successful current line 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 a previously semantically annotated corpus) have obtained better results than unsupervised methods on small sets of selected highly ambiguous words, or artificial pseudo–words. Many

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

R. López de Mántaras, E. Plaza (Eds.): ECML 2000, LNAI 1810, pp. 129–141, 2000.

standard ML algorithms for supervised learning have been applied, such as: Naive Bayes [19,22], [19,10], Exemplar–based learning Decision Lists [28], Neural Networks [27], etc. Further, Mooney [17] has also compared all previously cited methods on a very restricted domain and including Decision Trees and Rule Induction algorithms. Unfortunately, there have been very few direct comparisons of alternative methods on identical test data. However, it is commonly accepted that Naive Bayes, Neural Networks and Exemplar–based learning represent state–of–the–art accuracy on supervised WSD.

Supervised methods suffer from the lack of widely available semantically tagged corpora, from which to construct really broad coverage systems. This is known as the "knowledge acquisition bottleneck". Ng [20] estimates that the manual annotation effort necessary to build a broad coverage semantically annotated corpus would be about 16 man-years. This extremely high overhead for supervision and, additionally, the also serious overhead for learning/testing many of the commonly used algorithms when scaling to real size WSD problems, explain why supervised methods have been seriously questioned.

Due to this fact, recent works have focused on reducing the acquisition cost as well as the need for supervision in corpus–based methods for WSD. Consequently, the following three lines of research can be found: 1) The design of efficient example sampling methods [6,10]; 2) The use of lexical resources, such as WordNet [16], and WWW search engines to automatically obtain from Internet arbitrarily large samples of word senses [12,15]; 3) The use of unsupervised EM–like algorithms for estimating the statistical model parameters [22]. It is also our belief that this body of work, and in particular the second line, provides enough evidence towards the "opening" of the acquisition bottleneck in the near future. For that reason, it is worth further investigating the application of new supervised ML methods to better resolve the WSD problem.

Boosting Algorithms. The main idea of boosting algorithms is to combine many simple and moderately accurate hypotheses (called weak classifiers) into a single, highly accurate classifier for the task at hand. 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.

The AdaBoost.MH algorithm applied in this paper [25] is a generalization of Freund and Schapire's AdaBoost algorithm [9], which has been (theoretically and experimentally) studied extensively and which has been shown to perform well on standard machine–learning tasks using also standard machine–learning algorithms as weak learners [23,8,5,2].

Regarding Natural Language (NL) problems, AdaBoost.MH has been successfully applied to Part–of–Speech (PoS) tagging [1], Prepositional–Phrase–attachment disambiguation [1], and, Text Categorization [26] with especially good results.

The Text Categorization domain shares several properties with the usual settings of WSD, such as: very high dimensionality (typical features consist in testing the presence/absence of concrete words), presence of many irrelevant and highly dependent features, and the fact that both, the learned concepts and the examples, reside very sparsely in the feature space. Therefore, the application of AdaBoost.MH to WSD seems to be a promising choice. It has to be noted that, apart from the excellent results obtained on NL problems, AdaBoost.MH has the advantages of being theoretically well founded and easy to implement.

The paper is organized as follows: Section 2 is devoted to explain in detail the AdaBoost.MH algorithm. Section 3 describes the domain of application and the initial experiments performed on a reduced set of words. In Section 4 several alternatives are explored for accelerating the learning process by reducing the feature space. The best alternative is fully tested in Section 5. Finally, Section 6 concludes and outlines some directions for future work.

2 The Boosting Algorithm AdaBoost.MH

This section describes the Schapire and Singer's AdaBoost.MH algorithm for multiclass multi–label classification, using exactly the same notation given by the authors in [25,26].

As already said, the purpose of boosting is to find a highly accurate classification rule by combining many weak hypotheses (or weak rules), each of which may be only moderately accurate. It is assumed that there exists a separate procedure called the WeakLearner for acquiring the weak hypotheses. The boosting algorithm finds a set of weak hypotheses by calling the weak learner repeatedly in a series of T rounds. These weak hypotheses are then combined into a single rule called the combined hypothesis.

Let $S = \{(x_1, Y_1), \ldots, (x_m, Y_m)\}$ be the set of m training examples, where each instance x_i belongs to an instance space $\mathcal{X}$ and each Y_i is a subset of a finite set of labels or classes $\mathcal{Y}$. The size of $\mathcal{Y}$ is denoted by $k = |\mathcal{Y}|$.

The pseudo–code of AdaBoost.MH is presented in figure 1. AdaBoost.MH maintains an $m \times k$ matrix of weights as a distribution D over examples and labels. The goal of the WeakLearner algorithm is to find a weak hypothesis with moderately low error with respect to these weights. Initially, the distribution D_1 is uniform, but the boosting algorithm updates the weights on each round to force the weak learner to concentrate on the pairs (examples,label) which are hardest to predict.

More precisely, let D_t be the distribution at round t, and $h_t : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ the weak rule acquired according to D_t. The sign of $h_t(x, l)$ is interpreted as a prediction of whether label l should be assigned to example x or not. The magnitude of the prediction $|h_t(x, l)|$ is interpreted as a measure of confidence in the prediction. In order to understand correctly the updating formula this last piece of notation should be defined. Thus, given $Y \subseteq \mathcal{Y}$ and $l \in \mathcal{Y}$, let $Y[l]$ be +1 if $l \in Y$ and -1 otherwise.

Now, it becomes clear that the updating function increases (or decreases) the weights $D_t(i, l)$ for which h_t makes a good (or bad) prediction, and that this variation is proportional to $|h_t(x, l)|$.

Note that WSD is not a multi–label classification problem since a unique sense is expected for each word in context. In our implementation, the algorithm runs

procedure AdaBoost.MH (**in**: $S = \{(x_i, Y_i)\}_{i=1}^m$)
S is the set of training examples
Initialize distribution D_1 (for all i, $1 \le i \le m$, and all l, $1 \le l \le k$)
$D_1(i, l) = 1/(mk)$
for t:=1 **to** T **do**
Get the weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
$h_t = \mathsf{WeakLearner}(X, D_t)$;
Update distribution D_t (for all i, $1 \le i \le m$, and all l, $1 \le l \le k$)
$$D_{t+1}(i, l) = \frac{D_t(i, l)\exp(-Y_i[l]h_t(x_i, l))}{Z_t}$$
Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution)
end-for
return the combined hypothesis: $f(x, l) = \sum_{t=1}^{T} h_t(x, l)$
end AdaBoost.MH

Fig. 1. The AdaBoost.MH algorithm

exactly in the same way as explained above, except that sets Y_i are reduced to a unique label, and that the combined hypothesis is forced to output a unique label, which is the one that maximizes $f(x, l)$.

Up to now, it only remains to be defined the form of the WeakLearner. Schapire and Singer [25] prove that the Hamming loss of the AdaBoost.MH algorithm on the training set[1] is at most $\prod_{t=1}^{T} Z_t$, where Z_t is the normalization factor computed on round t. This upper bound is used in guiding the design of the WeakLearner algorithm, which attempts to find a weak hypothesis h_t that minimizes: $Z_t = \sum_{i=1}^{m} \sum_{l \in \mathcal{Y}} D_t(i, l)\exp(-Y_i[l]h_t(x, l))$.

2.1 Weak Hypotheses for WSD

As in [1], very simple weak hypotheses are used to test the value of a boolean predicate and make a prediction based on that value. The predicates used, which are described in section 3.1, are of the form "$f = v$", where f is a feature and v is a value (e.g.: "previous_word = *hospital*"). Formally, based on a given predicate p, our interest lies on weak hypotheses h which make predictions of the form:

$$h(x, l) = \begin{cases} c_{0l} & \text{if } p \text{ holds in } x \\ c_{1l} & \text{otherwise} \end{cases}$$

where the c_{jl}'s are real numbers.

For a given predicate p, and bearing the minimization of Z_t in mind, values c_{jl} should be calculated as follows. Let X_1 be the subset of examples for which the

[1] i.e. the fraction of training examples i and labels l for which the sign of $f(x_i, l)$ differs from $Y_i[l]$.

predicate p holds and let X_0 be the subset of examples for which the predicate p does not hold. Let $[\![\pi]\!]$, for any predicate π, be 1 if π holds and 0 otherwise. Given the current distribution D_t, the following real numbers are calculated for each possible label l, for $j \in \{0,1\}$, and for $b \in \{+1,-1\}$:

$$W_b^{jl} = \sum_{i=1}^{m} D_t(i,l)[\![x_i \in X_j \wedge Y_i[l] = b]\!]$$

That is, W_{+1}^{jl} (W_{-1}^{jl}) is the weight (with respect to distribution D_t) of the training examples in partition X_j which are (or not) labelled by l.

As it is shown in [25], Z_t is minimized for a particular predicate by choosing:

$$c_{jl} = \frac{1}{2}\ln\left(\frac{W_{+1}^{jl}}{W_{-1}^{jl}}\right)$$

These settings imply that:

$$Z_t = 2\sum_{j\in\{0,1\}} \sum_{l\in\mathcal{Y}} \sqrt{W_{+1}^{jl} W_{-1}^{jl}}$$

Thus, the predicate p chosen is that for which the value of Z_t is smallest.

Very small or zero values for the parameters W_b^{jl} cause c_{jl} predictions to be large or infinite in magnitude. In practice, such large predictions may cause numerical problems to the algorithm, and seem to increase the tendency to overfit. As suggested in [26], smoothed values for c_{jl} have been used.

3 Applying Boosting to WSD

3.1 Corpus

In our experiments the boosting approach has been evaluated using the DSO corpus containing 192,800 semantically annotated occurrences[2] of 121 nouns and 70 verbs. These correspond to the most frequent and ambiguous English words. The DSO corpus was collected by Ng and colleagues [18] and it is available from the Linguistic Data Consortium (LDC)[3].

For our first experiments, a group of 15 words (10 nouns and 5 verbs) which frequently appear in the related WSD literature has been selected. These words are described in the left hand–side of table 1. Since our goal is to acquire a classifier for each word, each row represents a classification problem. The number of classes (senses) ranges from 4 to 30, the number of training examples from 373 to 1,500 and the number of attributes from 1,420 to 5,181. The MFS column on the right hand–side of table 1 shows the percentage of the most frequent sense for each word, i.e. the accuracy that a naive "Most–Frequent–Sense" classifier would obtain.

The binary–valued attributes used for describing the examples correspond to the binarization of seven features referring to a very narrow linguistic context. Let "w_{-2} w_{-1} w w_{+1} w_{+2}" be the context of 5 consecutive words around the

[2] These examples are tagged with a set of labels which correspond, with some minor changes, to the senses of WordNet 1.5 [21].

[3] LDC e-mail address: `ldc@unagi.cis.upenn.edu`

word w to be disambiguated. The seven features mentioned above are exactly those used in [19]: w_{-2}, w_{-1}, w_{+1}, w_{+2}, (w_{-2}, w_{-1}), (w_{-1}, w_{+1}), and (w_{+1}, w_{+2}), where the last three correspond to collocations of two consecutive words.

3.2 Benchmark Algorithms and Experimental Methodology

AdaBoost.MH has been compared to the following algorithms:

Naive Bayes (NB). The naive Bayesian classifier has been used in its most classical setting [4]. To avoid the effect of zero counts when estimating the conditional probabilities of the model, a very simple smoothing technique has been used, which was proposed in [19].

Exemplar–based learning (EB_k). In our implementation, all examples are stored in memory and the classification of a new example is based on a k–NN algorithm using Hamming distance to measure closeness (in doing so, all examples are examined). If k is greater than 1, the resulting sense is the weighted majority sense of the k nearest neighbours (each example votes its sense with a strength proportional to its closeness to the test example). Ties are resolved in favour of the most frequent sense among all those tied.

The comparison of algorithms has been performed in series of controlled experiments using exactly the same training and test sets for each method. The experimental methodology consisted in a 10-fold cross-validation. All accuracy/error rate figures appearing in the paper are averaged over the results of the 10 folds. The statistical tests of significance have been performed using a 10-fold cross validation paired Student's t-test with a confidence value of: $t_{9,0.975} = 2.262$.

3.3 Results

Figure 2 shows the error rate curve of AdaBoost.MH, averaged over the 15 reference words, and for an increasing number of weak rules per word. This plot shows that the error obtained by AdaBoost.MH is lower than those obtained by NB and EB_{15} (k=15 is the best choice for that parameter from a number of tests between k=1 and k=30) for a number of rules above 100. It also shows that the error rate decreases slightly and monotonically, as it approaches the maximum number of rules reported[4].

According to the plot in figure 2, no overfitting is observed while increasing the number of rules per word. Although it seems that the best strategy could be "learn as many rules as possible", in [7] it is shown that the number of rounds must be determined individually for each word since they have different behaviours. The adjustment of the number of rounds can be done by cross–validation on the training set, as suggested in [1]. However, in our case, this cross–validation inside the cross–validation of the general experiment would generate a prohibitive overhead. Instead, a very simple stopping criterion (sc) has been

[4] The maximum number of rounds considered is 750, merely for efficiency reasons.

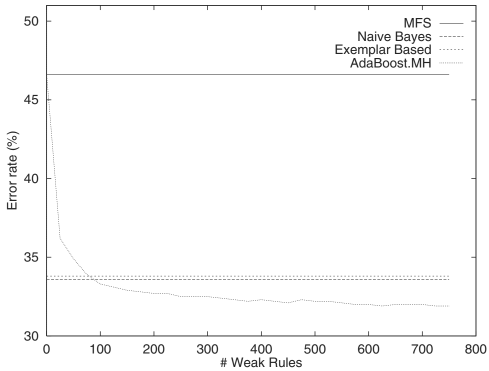

Fig. 2. Error rate of AdaBoost.MH related to the number of weak rules

used, which consists in stopping the acquisition of weak rules whenever the error rate on the training set falls below 5%, with an upper bound of 750 rules. This variant, which is referred to as AB_{sc}, obtained comparable results to AB_{750} but generating only 370.2 weak rules per word on average, which represents a very moderate storage requirement for the combined classifiers.

The numerical information corresponding to this experiment is included in table 1. This table shows the accuracy results, detailed for each word, of NB, EB_1, EB_{15}, AB_{750}, and AB_{sc}. The best result for each word is printed in boldface.

As it can be seen, in 14 out of 15 cases, the best results correspond to the boosting algorithms. When comparing global results, accuracies of either AB_{750} or AB_{sc} are significantly greater than those of any of the other methods. Finally, note that accuracies corresponding to NB and EB_{15} are comparable (as suggested in [19]), and that the use of k's greater than 1 is crucial for making Exemplar–based learning competitive on WSD.

4 Making Boosting Practical for WSD

Up to now, it has been seen that AdaBoost.MH is a simple and competitive algorithm for the WSD task. It achieves an accuracy performance superior to that of the Naive Bayes and Exemplar–based algorithms tested in this paper. However, AdaBoost.MH has the drawback of its computational cost, which makes the algorithm not scale properly to real WSD domains of thousands of words.

The space and time–per–round requirements of AdaBoost.MH are $\mathcal{O}(mk)$ (recall that m is the number of training examples and k the number of senses), not including the call to the weak learner. This cost is unavoidable since AdaBoost.MH is inherently sequential. That is, in order to learn the $(t+1)$-th weak rule it needs the calculation of the t-th weak rule, which properly updates the matrix D_t. Further, inside the WeakLearner, there is another iterative process that examines, one by one, all attributes so as to decide which is the one that

Table 1. Set of 15 reference words and results of the main algorithms

		Number of			Accuracy (%)					
Word	POS	Senses	Examp.	Attrib.	MFS	NB	EB_1	EB_{15}	AB_{750}	AB_{sc}
age	n	4	493	1662	62.1	73.8	71.4	71.0	**74.7**	74.0
art	n	5	405	1557	46.7	54.8	44.2	58.3	57.5	**62.2**
car	n	5	1381	4700	95.1	95.4	91.3	95.8	**96.8**	96.5
child	n	4	1068	3695	80.9	86.8	82.3	89.5	**92.8**	92.2
church	n	4	373	1420	61.1	62.7	61.9	63.0	**66.2**	64.9
cost	n	3	1500	4591	87.3	86.7	81.1	87.7	87.1	**87.8**
fall	v	19	1500	5063	70.1	76.5	73.3	79.0	**81.1**	80.6
head	n	14	870	2502	36.9	76.9	70.0	76.9	**79.0**	**79.0**
interest	n	7	1500	4521	45.1	64.5	58.3	63.3	**65.4**	65.1
know	v	8	1500	3965	34.9	47.3	42.2	46.7	**48.7**	**48.7**
line	n	26	1342	4387	21.9	51.9	46.1	49.7	**54.8**	54.5
set	v	19	1311	4396	36.9	**55.8**	43.9	54.8	**55.8**	**55.8**
speak	v	5	517	1873	69.1	**74.3**	64.6	73.7	72.2	73.3
take	v	30	1500	5181	35.6	44.8	39.3	46.1	**46.7**	46.1
work	n	7	1469	4923	31.7	**51.9**	42.5	47.2	50.7	50.7
Avg. nouns		8.6	1040.1	3978.5	57.4	71.7	65.8	71.1	**73.5**	73.4
verbs		17.9	1265.6	4431.9	46.6	57.6	51.1	58.1	**59.3**	59.1
all		12.1	1115.3	4150.0	53.3	66.4	60.2	66.2	**68.1**	68.0

minimizes Z_t. Since there are thousands of attributes, this is also a time consuming part, which can be straightforwardly spedup either by reducing the number of attributes or by relaxing the need to examine all attributes at each iteration.

4.1 Accelerating the WeakLearner

Four methods have been tested in order to reduce the cost of searching for weak rules. The first three, consisting in aggressively reducing the feature space, are frequently applied in Text Categorization. The fourth consists in reducing the number of attributes that are examined at each round of the boosting algorithm.

Frequency filtering (Freq): This method consists in simply discarding those features corresponding to events that occur less than N times in the training corpus. The idea beyond that criterion is that frequent events are more informative than rare ones.

Local frequency filtering (LFreq): This method works similarly to Freq but considers the frequency of events locally, at the sense level. More particularly, it selects the N most frequent features of each sense.

RLM ranking: This third method consists in making a ranking of all attributes according to the RLM distance measure [13] and selecting the N most relevant features. This measure has been commonly used for attribute selection in decision tree induction algorithms[5].

[5] RLM distance belongs to the distance–based and information–based families of attribute selection functions. It has been selected because it showed better perfor-

LazyBoosting: The last method does not filter out any attribute but reduces the number of those that are examined at each iteration of the boosting algorithm. More specifically, a small proportion p of attributes are randomly selected and the best weak rule is selected among them. The idea behind this method is that if the proportion p is not too small, probably a sufficiently good rule can be found at each iteration. Besides, the chance for a good rule to appear in the whole learning process is very high. Another important characteristic is that no attribute needs to be discarded and so we avoid the risk of eliminating relevant attributes[6].

The four methods above have been compared for the set of 15 reference words. Figure 3 contains the average error–rate curves obtained by the four variants at increasing levels of attribute reduction. The top horizontal line corresponds to the MFS error rate, while the bottom horizontal line stands for the error rate of AdaBoost.MH working with all attributes. The results contained in figure 3 are calculated running the boosting algorithm 250 rounds for each word.

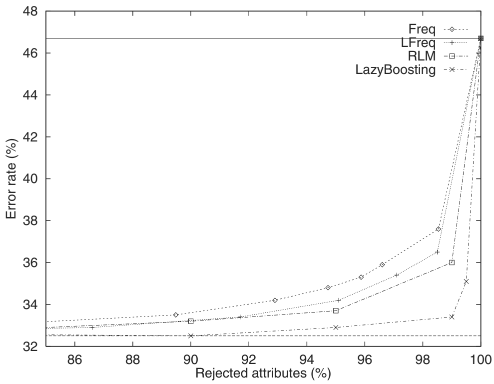

Fig. 3. Error rate obtained by the four methods, at 250 weak rules per word, with respect to the percentage of rejected attributes

The main conclusions that can be drawn are the following:

- All methods seem to work quite well since no important degradation is observed in performance for values lower than 95% in rejected attributes. This may indicate that there are many irrelevant or highly dependent attributes in our domain.

---

mance than seven other alternatives in an experiment of decision tree induction for PoS tagging [14].

[6] This method will be called LazyBoosting in reference to the work by Samuel and colleagues [24]. They applied the same technique for accelerating the learning algorithm in a Dialogue Act tagging system.

- LFreq is slightly better than Freq, indicating a preference to make frequency counts for each sense rather than globally.
- The more informed RLM ranking performs better than frequency–based reduction methods Freq and LFreq.
- LazyBoosting is better than all other methods, confirming our expectations: it is worth keeping all information provided by the features. In this case, acceptable performance is obtained even if only 1% of the attributes is explored when looking for a weak rule. The value of 10%, for which LazyBoosting still achieves the same performance and runs about 7 times faster than AdaBoost.MH working with all attributes, will be selected for the experiments in section 5.

5 Evaluating LazyBoosting

The LazyBoosting algorithm has been tested on the full semantically annotated corpus with $p = 10\%$ and the same stopping criterion described in section 3.3, which will be referred to as AB_{l10sc}. The average number of senses is 7.2 for nouns, 12.6 for verbs, and 9.2 overall. The average number of training examples is 933.9 for nouns, 938.7 for verbs, and 935.6 overall.

The AB_{l10sc} algorithm learned an average of 381.1 rules per word, and took about 4 days of CPU time to complete[7]. It has to be noted that this time includes the cross–validation overhead. Eliminating it, it is estimated that 4 CPU days would be the necessary time for acquiring a word sense disambiguation boosting–based system covering about 2,000 words.

The AB_{l10sc} has been compared again to the benchmark algorithms using the 10-fold cross–validation methodology described in section 3.2. The average accuracy results are reported in the left hand–side of table 2. The best figures correspond to the LazyBoosting algorithm AB_{l10sc}, and again, the differences are statistically significant using the 10-fold cross–validation paired t-test.

Table 2. Results of LazyBoosting and the benchmark methods on the 191–word corpus

	Accuracy (%)				Wins–Ties–Losses	
	MFS	NB	EB_{15}	AB_{l10sc}	AB_{l10sc} vs. NB	AB_{l10sc} vs. EB_{15}
Nouns (121)	56.4	68.7	68.0	**70.8**	99(51) – 1 – 21(3)	100(68) – 5 – 16(1)
Verbs (70)	46.7	64.8	64.9	**67.5**	63(35) – 1 – 6(2)	64(39) – 2 – 4(0)
Average (191)	52.3	67.1	66.7	**69.5**	162(86) – 2 – 27(5)	164(107) – 7 – 20(1)

The right hand–side of the table shows the comparison of AB_{l10sc} versus NB and EB_{15} algorithms, respectively. Each cell contains the number of wins,

[7] The current implementation is written in PERL-5.003 and it was run on a SUN UltraSparc2 machine with 194Mb of RAM.

ties, and losses of competing algorithms. The counts of statistically significant differences are included in brackets. It is important to point out that EB_{15} only beats significantly AB_{l10sc} in one case while NB does so in five cases. Conversely, a significant superiority of AB_{l10sc} over EB_{15} and NB is observed in 107 and 86 cases, respectively.

6 Conclusions and Future Work

In the present work, Schapire and Singer's AdaBoost.MH algorithm has been evaluated on the word sense disambiguation task, which is one of the hardest open problems in Natural Language Processing. As it has been shown, the boosting approach outperforms Naive Bayes and Exemplar–based learning, which represent state–of–the–art accuracy on supervised WSD. In addition, a faster variant has been suggested and tested, which is called LazyBoosting. This variant allows the scaling of the algorithm to broad-coverage real WSD domains, and is as accurate as AdaBoost.MH. Further details can be found in an extended version of this paper [7].

Future work is planned to be done in the following directions:

- Extensively evaluate AdaBoost.MH on the WSD task. This would include taking into account additional attributes, and testing the algorithms in other manually annotated corpora, and especially on sense–tagged corpora automatically obtained from Internet.
- Confirm the validity of the LazyBoosting approach on other language learning tasks in which AdaBoost.MH works well, e.g.: Text Categorization.
- 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 boosting as a way to improve data quality [26,1]. It is suspected that the corpus used in the current work is very noisy, so it could be worth using boosting to try and improve it.

References

1. Abney, S., Schapire, R.E. and Singer, Y.: *Boosting Applied to Tagging and PP–attachment*. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999. 130, 132, 134, 139
2. Bauer, E. and Kohavi, R.: *An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants*. Machine Learning Journal. Special issue on IMLM for Improving and Scaling Machine Learning Algorithms, 1999. 130
3. Breiman, L.: *Arcing Classifiers*. The Annals of Statistics, 26(3), 1998.
4. Duda, R. O. and Hart, P. E.: *Pattern Classification and Scene Analysis*. Wiley, New York, 1973. 134

5. Dietterich, T.G.: *An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization.* Machine Learning (to appear). 130
6. Engelson, S.P. and Dagan, I.: *Minimizing Manual Annotation Cost in Supervised Training from Corpora.* In S. Wermter, E. Riloff and G. Scheler, editors, Connectionist, Statistical an Symbolic Approaches to Learning for Natural Language Processing, LNAI, 1040. Springer, 1996. 130
7. Escudero, G., Màrquez, L. and Rigau, G. *Boosting Applied to Word Sense Disambiguation.* Technical Report LSI-00-3-R, LSI Department, UPC, 2000. 134, 139
8. Freund, Y. and Schapire, R.E.: *Experiments with a New Boosting Algorithm.* In Procs. of the 13th International Conference on Machine Learning, ICML, 1996. 130
9. Freund, Y. and Schapire, R.E.: *A Decision–theoretic Generalization of On-line Learning and an Application to Boosting.* Journal of Computer and System Sciences, 55(1), 1997. 130
10. Fujii, A., Inui, K., Tokunaga, T. and Tanaka, H.: *Selective Sampling for Example–based Word Sense Disambiguation.* Computational Linguistics, 24(4), ACL, 1998. 130
11. Ide, N. and Véronis, J.: *Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art.* Computational Linguistics, 24(1), ACL, 1998. 129
12. Leacock, C., Chodorow, M. and Miller, G.A.: *Using Corpus Statistics and WordNet Relations for Sense Identification.* Computational Linguistics, 24(1), ACL, 1998. 130
13. López de Mántaras, R.: *A Distance–based Attribute Selection Measure for Decision Tree Induction.* Machine Learning, 6(1), 1991. 136
14. Màrquez, L.: *Part-of-speech Tagging: A Machine Learning Approach based on Decision Trees.* Phd. Thesis, LSI Department, UPC, 1999. 137
15. Mihalcea, R. and Moldovan, I.: *An Automatic Method for Generating Sense Tagged Corpora.* In Proceedings of the 16th National Conference on Artificial Intelligence, AAAI, 1999. 130
16. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K.: *Five Papers on WordNet.* Special Issue of International Journal of Lexicography, 3(4), 1990. 130
17. Mooney, R.J.: *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, 1996. 130
18. Ng, H.T. and Lee, H.B.: *Integrating Multiple Knowledge Sources to Disambiguate Word Senses: An Exemplar-based Approach.* In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, ACL, 1996. 133
19. Ng, H.T.: *Exemplar-based Word Sense Disambiguation: Some Recent Improvements.* In Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing, EMNLP, 1997. 130, 134, 135
20. Ng, H.T.: *Getting Serious about Word Sense Disambiguation.* In Proceedings of the SIGLEX Workshop "Tagging Text with Lexical Semantics: Why, What and How?", 1997. 130
21. Ng, H.T., Chung, Y. L. and Shou, K. F.: *A Case Study on Inter-Annotation Agreement for WSD.* In Proceedings of the SIGLEX Workshop "Standardizing Lexical Resources", Maryland, USA, 1999. 133

22. Pedersen, T. and Bruce, R.: *Knowledge Lean Word-Sense Disambiguation*. In Proceedings of the 15th National Conference on Artificial Intelligence, 1998. 130
23. Quinlan, J.R.: *Bagging, Boosting and C4.5*. In Proceedings of the 13th National Conference on Artificial Intelligence, AAAI, 1996. 130
24. Samuel, K.: *Lazy Transformation-Based Learning*. In Proceedings of the 11th International Florida AI Research Symposium Conference, 1998. 137
25. Schapire, R.E. and Singer, Y.: *Improved Boosting Algorithms Using Confidence-rated Predictions*. Machine Learning (to appear). 130, 131, 132, 133
26. Schapire, R.E. and Singer, Y.: *BoosTexter: A Boosting-based System for Text Categorization*. Machine Learning (to appear). 130, 131, 133, 139
27. Towell, G. and Voorhees, E.M.: *Disambiguating Highly Ambiguous Words*. Computational Linguistics, 24(1), ACL, 1998. 130
28. Yarowsky, D.: *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, ACL, 1994. 130